Udipto Chowdhury
Prof. Jin Woo Kim
CSCI 374
12/15/2021

# Homework 8

3c. Rewrite the following code segment using a multiple-selection statement

- ```
  #include <iostream>
  using namespace std;
  int main ()
  {
      int k;
      cout << "Enter your k value:";
      cin >> k >> \n;
      return 0;
  }

  {
      if ((k == 1) || (k == 2)) {
      cout << "Here's your j value:" << 2*k-1;
  }
      else if ((k == 3) || (k == 5)) {
      cout << "Here's your j value:" << 3*k+1;
  }
      else if (k == 4) {
      cout << "Here's your j value:" << 4*k-1;
  }
      else ((k == 6) || (k == 7) || (k == 8)) {
      cout << "Here's your j value:" << k-2;
  }
  ```
  ```

4. Rewrite the code without goto's or breaks:

```
j = -3;
for (i=0; i < 3; i++) {
        switch (j + 2) {
                case 0: j += 2;
                case 1: j = 0
                case 2: j--;
        }
        if (j > 0) {
        j = 3-i
}
```

★ Written Portion

- What are the pros and cons of using unique closing reserved words on compound statements?
  - Unique closing keywords on compound statements on one hand have the advantage of readability and on the other hand have the disadvantage of complicating the language by increasing the number of keywords.
- What are the arguments both for and against the exclusive use of Boolean expressions in the control statements in Java (as opposed to also allowing arithmetic expressions, as in C++)?
  - The primary argument for using Boolean expressions exclusively as control expressions is the reliability that results from not allowing a wide range of types for this use. In C++ for instance, an expression of any type can appear as a control expression, so typing errors that result in references to variables of incorrect types are not detected by the compiler as errors. The disadvantage, on the other hand, is the writability. And the reason the writability is a disadvantage is because it is hard to complicate a programming language (C++ in this case) if the keywords' limit is increased. Using more of these reserved words means less words available for the programmer.

Homework 9

★ In most Fortran IV implementations, parameters were passed by reference, using access path transmission only. State both the advantages and disadvantages of this design choice.
  - Advantages: C++ provides a convenient and effective method for specifying write protection on pass-by-value parameters, the passing process itself is efficient – in terms of both time and space, duplicate space is not required and no copying is required.
  - Disadvantages: Access to the formal parameters will be slower than pass-by-value parameters, because of the additional level of indirect addressing that is required. If only one-way communication to the called subprogram is required, inadvertent and erroneous changes may be made to the actual parameter. Another problem of pass-by-reference is that aliases can be created. Because pass- by- reference makes access paths available to the called subprograms, thereby providing access to nonlocal variables which makes program verification more difficult.

★ Consider the following program written in C syntax:

```
void swap(int a, int b) {
  int temp;
  temp = a;
  a = b;
  b = temp;
}
void main() {
  int value = 2, list[5] = {1, 3, 5, 7, 9};
  swap(value, list[0]);
  swap(list[0], list[1]);
  swap(value, list[value]);
}
```

  - For each of the following parameter-passing methods, what are all of the values of the variables value and list after each of the three calls to swap?
    - Pass By Value
      - With pass by value, nothing changed. Therefore, the defined variables are unchanged.

- Pass By Reference
  - With pass by reference, the arguments get changed. How? Since list[0] = 1 and value is 2 by initial value, the values are being swapped (list[0] = 2 & int value is 1). If we take a look at the 10th code line, because the 9th line already defines list[0], we know list[1] is 3 by definition (list[1] = 2 & list[0] = 3). Finally, we know that the value is 1 based on our first step. So, list[value] = list[1] = 2 by definition from the 10th line. Therefore, if we swap it the value finally returns to its original value (which is 2) and list[1] is 1.
- Pass By Value Result
  - (Refer to Reference since it is pretty much the same thing)

★ 7. Consider the following program written in C syntax:

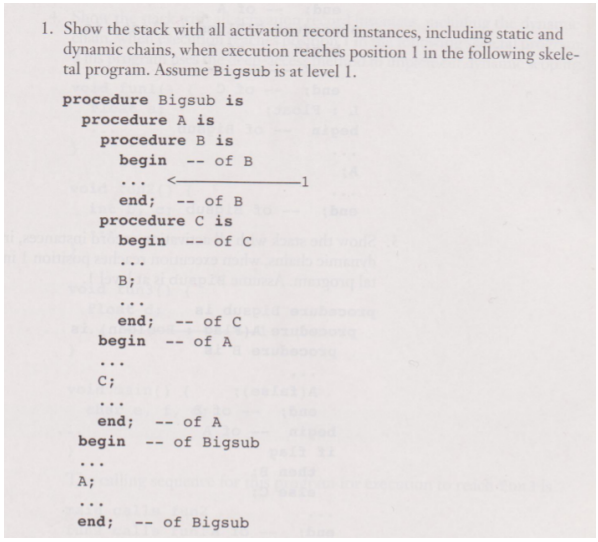
```
void fun (int first, int second) {
  first += first;
  second += second;
}
void main() {
  int list[2] = {1, 3};
  fun(list[0], list[1]);
}
```

For each of the following parameter-passing methods, what are the values of the `list` array after execution?

- Passed By Value
  - 1 and 3 by initial value from the list of two numbers.
- Passed By Reference
  - By executing the function, 1+1 = 2 and 3*2 = 3+3 = 6. Therefore, the values will be 2,6 ((1,3) * 2)
- Passed By Value-Result
  - (Again, same thing ; does that mean ‘value-result’ == ‘reference’?)

Homework 10

★ Show the stack with all Activation Record Instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume **Bigsub** is at level 1.



1. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume Bigsub is at level 1.

```
procedure Bigsub is
  procedure A is
    procedure B is
      begin  -- of B
      ...  <-------------1
      end;  -- of B
    procedure C is
      begin  -- of C
      ...
      B;
      ...
      end;  -- of C
    begin  -- of A
    ...
    C;
    ...
    end;  -- of A
  begin  -- of Bigsub
  ...
  A;
  ...
  end;  -- of Bigsub
```

★
- Doing this in descending order
  - Activation Record Instances for B
    - Dynamic Link
    - Static Link
    - Return to C
  - Activation Record Instances for C
    - Dynamic Link
    - Static Link
    - Return to A
  - Activation Record Instances for A
    - Dynamic Link
    - Static Link
    - Return to **Bigsub**

        - Activation Record Instances for **Bigsub**
            - Dynamic Link
            - Static Link
            - Return (Return 0 for C++)
            - -
            - -
            - Stack
    - Explanation: Based on the code, if the specific pair of begin and end goes first, that variable is on the top floor (in this case lines 4 & 6 is indicating that ‘B’ is first one to do an ARI.

★ 


- Since there is a 1 between C-pairs (between begin and end ; C's ARI will be on top floor)
  - Activation Record Instances for C
    - Dynamic Link
    - Static Link
    - Return to B (B pairs are surrounded by C(Z))

- Activation Record Instances for B
  - Parameter (defined variable C(Z))
  - Dynamic Link
  - Static Link
  - Return to A (Look for a pair that surrounds ‘B’; in this case, it’s ‘A’)
- Activation Record Instances for A
  - Parameters (defined variable B(X) and summation function)
  - Dynamic Link
  - Static Link
  - Return to **Bigsub**
- Activation Record Instances for **Bigsub**
  - Dynamic Link
  - Static Link
  - Return (MySum)
  - -
  - -
  - Stack

★ It is stated in the chapter that when nonlocal variables are accessed in a dynamic-scoped language using the dynamic chain, variable names must be stored in the activation records with the values. If this were actually done, every non-local access would require a sequence of costly string comparisons on names. Design an alternative to these string comparisons that would be faster.

- The best alternative for string comparisons is to assign distinct (specific) integer values to all variables on the program. Also, use these integer values instead of a generic string variable name for the activation records. This way, the comparisons will be more efficient between two integer values (numeric values) since characters carry large hexadecimal values when it comes to string comparison.

★ Pascal allows gotos with nonlocal targets. How could such statements be handled if static chains were used for nonlocal variable access?

- When a reference is made to a nonlocal variable, the activation record instance (ARI) contains the variable. This can be found by searching up the static chain until a static ancestor activation record instance is found that contains that variable.

★ The static-chain method could be expanded slightly by using two static links in each activation record instance where the second points to the static grandparent activation record instance. How would this approach affect the time required for subprogram linkage and nonlocal references?

- To approach this situation, we have to include two static links which will reduce the access time to non-locals that are defined in scopes. In order to equate well with each other, the scopes need to be two steps away and non-locals need to be a step away. With that in mind, because most non-local references are relatively close, it could significantly increase the execution efficiency of many executable programs.