PF Spring 2025: Code Dry Run Examples

C++ Code Dry Run

- Simulates Line by Line Execution of C++ Code
- Dry Run starts from the very first executable line of main
- All variables must be shown with rectangle to indicate memory:

Data Type

* address | Variable value | variable name

- Address can be indicated as a1 or ad1 or adds1 or address1
- Addresses must be consecutive: ad1, ad2, ….
- **For cin**: a value will be given, assumed to be entered by the user
- **For cout**: show what will be displayed on screen
- **For if-else:** first evaluate the condition and then follow the flow
- **For loops:** first evaluate the condition and then continue or stop the loop. Must show all iterations of the loop using a table (see below for examples)
- **For array:**
  - All elements of the array are shown together as multiple cells (see examples)
  - The value of array, address, is passed to function and no local variables are created, instead the array address gets the name from the function. Although only one address is passed to function, the rest of the array exists in memory and can be used and changed. Al changes made to memory in a function remain in the memory. **You can't delete any variable, but can changes its value. Once a variable (and array) is created, it exists in memory until it goes out of scope.**

Code 1:

```
1.  double sum(double x, double y) // 2 local variables x and y are created to receive values
2.  {
3.      double result = x + y; // 2.5 + 3.5 = 6.0 is saved in variable result
4.      return result;  // values of result 6.0 is returned from function
5.  }  // all local variables (x, y, result) are deleted
6.  int main()
7.  {
8.      double a = 2.5;
```

double — ad3 | 2.5 | ad4

double — 2.5

double — ad5 | 6.0 | result

double — ad1 | 2.5 | a

double — 3.5

```
9.      double b = 3.5;          ad2                          b

10. double c = sum(a, b);  // Right Hand Side executes first
    // Left Hand Side is executed after functions call
    // variable c is created to hold the returned values from the function
                        double
            ad3   [ 6.0 ]   c
    // call to function sum, with 2.5 and 3.5 passed as parameters
11.     cout << a << " + " << b << " = " << c << endl; // show that will be displayed on screen
    // 2.5 + 3.5 = 6.0
12. return 0;  // Program ends
13. }
```

Code 2:

```
1.  int toDigit(char c) // local variable receives the value
                                  char
                        ad2   [ '6' ]   c
2.  {
3.      int result = 0;                int
                        ad3   [ 0 ]   result


4.      if(c <= '9' && c >= '0') // replace variables with values and evaluate condition
    //  '6' <= '9 && '6' >= '0' is true so go in the body of if
5.      {
6.          result = c – 48; // character is converted to an integer
                              int
                ad3   [ 6 ]   result
7.      }
8.      return result; // value of result 6 is returned
9.  } // delete local variables: c and result
10. int main()
11. {
12.     char a = '6';                    char
                        ad1   [ '6' ]   a
13.     int x = toDigit(a); // Right Hand Side, function call, is executed first
        // value of a '6' is passed to function toDigit
        // Left hand Side execution creates a variable x to hold the returned value from the
function                        int
            ad2   [ 6 ]   x


14.     cout << x << " " << c << endl; // shows the display on screen
15. // 6 6
16.     return 0; // program terminates
```

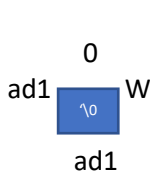
```
17. }
```

Code 3:

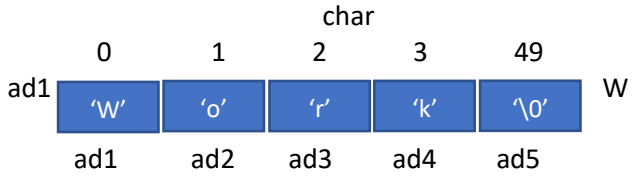
```
1.  double sum(double v[]) // function receives an address for v, no local variables are
    created, just the name of the memory address becomes v
                    double
            ad1     6       v
    // the memory at address ad2 exists and contains 3.5
    // v[0] is variable at ad1 and v[1] is variables at ad2

2.  {
3.      double result = 0.0
                    double
            ad3     0.0     result

4.      result = v[0] + v1[1]; // v[0] has 2.5 and v[1] has 3.5
                    double
            ad3     6.0     result

5.      return result; // value 6.0 of result  is returned
6.  } // local variable result is deleted
7.  int main()
8.  {
9.      double ar[] = {2.5, 3.5}; // 2 variables of type double are created
                                double
                            0       1
            ad1         2.5     3.5     ar
                        ad1     ad2
10.     double c = sum(ar);  // Right Hand Side Executes First
        // values of ar, address ad1, is passed to function sum
    // For :Left hand side, variable c is created to received the returned value from function
    toDigit
                    double
            ad3     6.0     c
11.     cout << ar[0] << " + " << ar[1] << " =  " << c << endl; // display on screen
    // 2.5 + 3.5 = 6.0
12.     return 0;
13. }
```

Code 4:

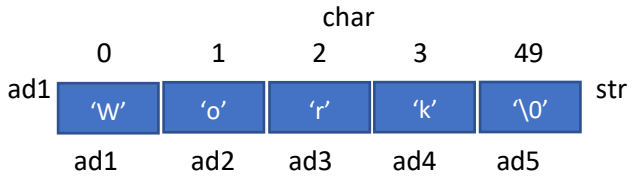
```
1.  void read(char W[]) // Memory at address ad1 is now named W
    // memory from ad1 to ad10 is available in memory
```



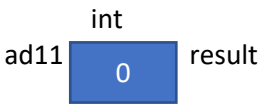
2. {
3. cout << “Enter a word: “; // show the display
   // Enter a word:
4. cin >> W; // assume user enters “Work”
   // cin reads the whole cstring in memory starting with address ad1



   // this memory will retain the values after the function
5. } // function returns
6. int length(char str[]) // the memory at ad1 is now called str and has the contents



7. {
8. int result = 0;

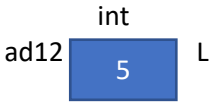


9. while(str[result++] != ‘\0’); // the loop dry run is done in table, where each iteration of the loop is shown

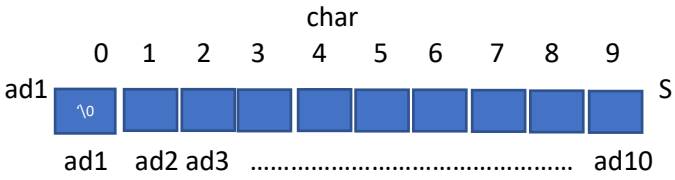
result	Variable or Expression	Value	result++
0	str[0]	‘W’	1
	str[0] != ‘\0’	‘W’ != ‘\0’ (true)	
1	str[1]	‘o’	2
	str[1] != ‘\0’	‘o’ != ‘\0’ (true)	
2	str[2]	‘r’	3
	str[2] != ‘\0’	‘r’ != ‘\0’ (true)	
3	str[3]	‘k’	4
	str[3] != ‘\0’	‘k’ != ‘\0’ (true)	
4	str[3]	‘\0’	5
	str[3] != ‘\0’	‘\0’ != ‘\0’ (false)	

10. return result; // return result value 5
11. } // delete local variable result

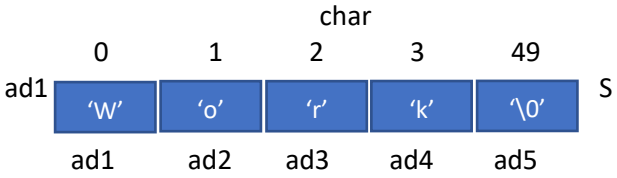
12. void display(char str[], int L)// address ad1 now has name str and a local variable is created L to receive value of 5



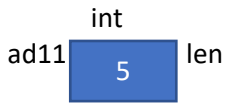
13. {
14. cout << "Length of " << str << " is " << L << endl; // show as displayed
15. // Length of Work is 5
16. } delete local variable L and return
17. int main()
18. {
19. char S[10] = ""; // 10 char variables with first equal to '\0' rest unknown



20. read(S); // Value of S i.e. address ad1 of S is passed to function S
// After the function execution, the memory from ad1 retains the values put in during the function, but now the array is named S



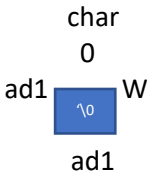
21. int len = length(S); // Right Hand Side will Execute first and function length is called with array S passed i.e. address of S ad1
// left hand side: create a variable len to received the returned value from length



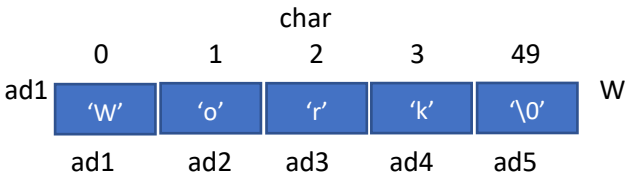
22. display(S, len); // array S addess ad1 and value of len 5 is passed to function display
23. return 0; // program ends
24. }

Code 5:

1. void read(char W[])
// memory from ad1 to ad10 is available in memory



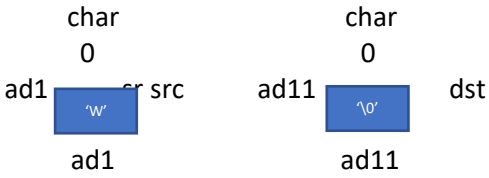
2. {
3. cout << "Enter a word: "; // show the display
   // Enter a word:
4. cin >> W; // assume user enters "Work"
   // cin reads the whole cstring in memory starting with address ad1



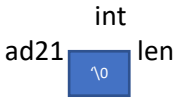
   // this memory will retain the values after the function
5. } // function returns

6. void copy(char src[], char dst) // ad1 is now named src and ad11 is named dst

   // memory from ad1 to ad10 and from ad11 to ad20 is available in memory



7. {
8. int len = 0;

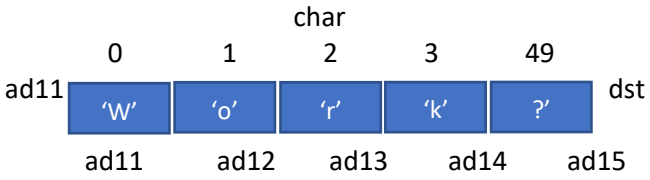


9. while(src[len] != '\0') // loop iteration is shown in a table
10. {
11. dst[len] = src[len];

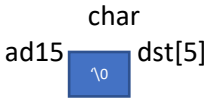
12. len++;
13. }

Line #	len	Variable or Expression	Value
9	0	src[len]	'W'
9	0	src[len] != '\0'	True
11	0	dst[len] = src[len]	dst[0] = src[0] = 'W'
12	0	len++	1
9	1	src[len]	'o'
9	1	src[len] != '\0'	True
11	1	dst[len] = src[len]	dst[1] = src[1] = 'o'
12	1	len++	2
9	2	src[len]	'r'
9	2	src[len] != '\0'	True
11	2	dst[len] = src[len]	dst[0] = src[0] = 'r'
12	2	len++	3
9	3	src[len]	'k'
9	3	src[len] != '\0'	True
11	3	dst[len] = src[len]	dst[0] = src[0] = 'k'
12	3	len++	4
9	4	src[len]	'\0'
9	4	src[len] != '\0'	False
		len++	5 (loop terminates)

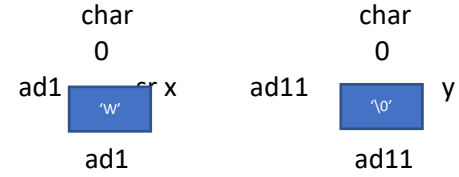
// after the loop is done here is the situation in memory of dst



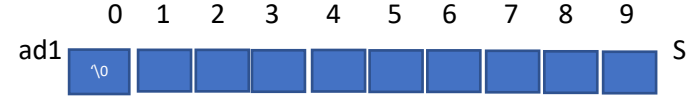
14. dst[len] = '\0'; // makes dst a cstring



15. } // deletes local variable len and returns with memory changes from ad11 to ad15
16. void display(char x[], char y[]) // address ad1 is now a and ad11 is y

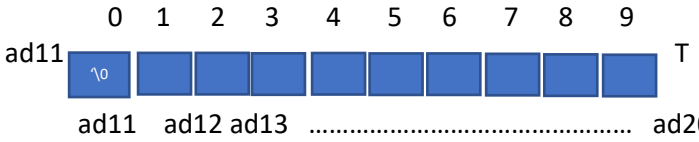


17. {
18. cout << "Copy of " << x << " is " << y << endl; // show as displayed
// cout keeps displaying characters from ad1 and ad11 ill finds the character '\0'
// Copy of Work is Work
19. }
20. int main()
21. {
22. char S[10] = ""; // 10 variables of type char, first value init to '\0'

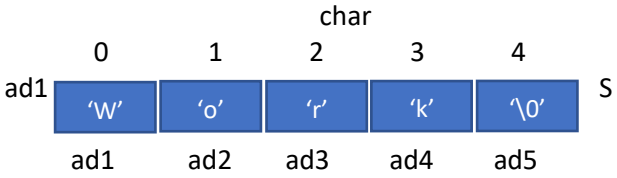


ad1 ad2 ad3 …… ad10

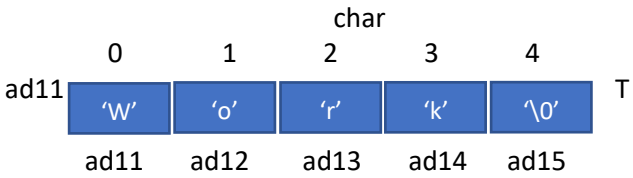
23. char T[10] = “”;// 10 variables of type char, first value init to ‘\0’



24. read(S); // S passes (ad1) to function read
// After the function execution, the memory from ad1 retains the values put in during the function, but now the array is named S



25. copy(S, T); // arrays S, address ad1 and T, address ad11 are passed to function copy
// memory changes will remain in address ad11 to ad15



26. display(S, T); S (address ad1) and T 9address ad11) are passed to function display
27. return 0;
28. } // all local variables including arrays are deleted and program terminates