Dayata, Wayne Matthew A. 1/26/2021

CIS 1201 (Group 1) Assignment #1 – Drawing execution stacks accurately

**FUNCTION SPECIFICATION:** The function compute will accept as parameters two valus A and B. If A is greater than B, the function will multiply A and B, and put the product in A; otherwise, the function will put the sum of A and B in A.

1. Function Header: void compute (int A, int B)

2. Function Call: int num1 = 10;

int num2 = 10;

compute (num1, num2);

3. Function code:

void compute (int \*A, int B) {

if (\*A > B) {  
 \*A \*= B;

} else {

\*A += B;

}

}

4. Execution Stack (with updates) (assume function is called in main() ):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | A |  | B | | **A100** |  | **10** | | B100 |  | B200 | | Activation record of compute():  **void compute (int \*A, int B) {**  **if (\*A > B) { /\* false \*/  \*A \*= B;**  **} else { /\* compiler proceeds here \*/**  **\*A += B;**  **}**  **}** |
| |  |  |  | | --- | --- | --- | | num1 |  | num2 | | **~~10~~ 20** |  | **10** | | A100 |  | A200 | | Activation record of main():  **int num1 = 10;**  **int num2 = 10;**  **compute (&num1, num2);** |

Things to note: The original values in main () are not manipulated, only the values in the function compute() are updated. All values are passed by copy.

int x = 10;

int y = 10;

compute(&x,y);

Feb 1, 2021

|  |  |  |  |
| --- | --- | --- | --- |
| Arr |  |  |  |
| A100 | 0 |  | A100-A103 |
|  | 1 |  | A104-A107 |
|  | 2 |  |  |
|  | 3 |  |  |
|  | 4 |  |  |
|  | 5 |  |  |
|  | 6 |  |  |
|  | 7 |  |  |

If you pass an array, you write Arr as parameter.

Arr holds the address of the first element of the array.

In other words, Arr == &A[0]

Arr[1] = \*(Arr+1)

printf(“%d”,sizeof(ptr));

printf(“%s”,\*ptr);

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | e | l | l | o | \0 | C | h | e | e | r | s | \0 |

S1 S2

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| G | o | o | d |  | d | a | y | \0 |  |  |  |  |

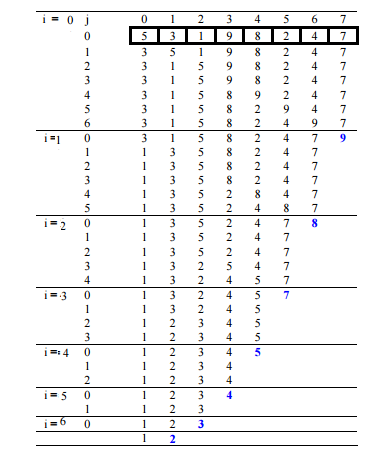
S3

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| G | o | o | d |  | d | a | y | \0 | e | r | s | \0 |

S1 S2

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| G | o | o | d |  | d | a | y | \0 |  |  |  |  |

S3



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Str1 |  | Str2 |  | Str3 |
| 0 |  | 0 | ‘H’ | 0 | ‘H’ |
| 1 |  | 1 | ‘E’ | 1 | ‘I’ |
| 2 |  | 2 | ‘L’ | 2 | 0 |
| 3 |  | 3 | ‘L’ | 3 | 0 |
| 4 |  | 4 | ‘O’ | 4 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | p | B |  |  |
|  | F100 | F100 | F100-F103 | 100 |
|  | G100 |  | F104-F107 | 200 |
|  |  |  | F108-F10B | 300 |
|  |  |  | F10C-F10F | 400 |
|  |  |  | F110-F113 | 0 |

int \*p;

p=B;

1. F100

2. 100

3. 200

4. F104

5. 100

6. B[0]

7. \*B, \*p

8. \*(B+1) \*(p+1)

9. yes

10. F104

struct Product{

char fname[24];

char MI;

char lname[16];

};

struct Cartesian Point;

struct Name Student;

struct ProductCode Product;

Point.x = 0;

Point.y = 0;

Name.fname = ”Juan”; //use strcpy

Name.MI = ‘T’;

Name.lname = ”dela Cruz”; //use strcpy

strcpy(Product.ProdDesc, “Hershey’s Chocolate 100 gms.”);

Product.ProdPrice = 149.50;