#include <stdio.h>

#define SIZE 4

Int foo(int \*a)

{

Return \*a+4;

}

Int foo1(char \* w[])

{

Int i;

for(i=0;i<SIZE;i++)

{

printf("%s", w[i]);

}

return i;

}

Int main (int argc, char\*\*argv)

{

Int continue1=8;

int num, num2;

int \*ptr=&num;

char \* words[]={"one", "two", "three", "four"};

while(continue1)

{

printf("\n\nEnter a number: 1, 2 or 3 to quit: \n");

scanf("%d", ptr);

if(num==1)

{

printf("Enter a number: ");

scanf("%d", ptr);

num2=foo(ptr);

printf("Number returned: %d\n", num2);

}

Else if(\*ptr==2)

{

num2=foo1(words);

printf("\nNumber returned:%d\n", num2);

}

Else if(\*ptr==3)

{

continue1=0;

}

else

{

printf("Not a valid entry.\n");

}

}

}

1. Yes, The function foo1 will always return the same value because the value size is always the same.
2. No, The initial value of ptr is the address value of

3. No, there is another way that the program exists and that is by entering a value one two three.

4. True, the first is printing out the address value and so is the second.

5. False, this is because the value that is entered into it is the address of the value of number entered by the user.

6. True, this is because ptr holds the address value of num so if you dereferenced it then it would mean that its going back to the original value. So that is the same as the value of num.

7. true, the loop runs as long as there is a variable continue1.

8. False, Foo and foo1 have different parameters.

9. True, because the variable ptr is holding the address value of an integer.

10. False, another unary operator used in this code is the increment operator ++.

11. False, foo and foo1 paraemets have different bytes because they are different in their formant. One is an integer which holds 4 bytes while the other is a char which doesn’t hold 4 bytes.

12. True, In foo1 the statement will output the address of each letter of the for loop iteration.

13. Yes, in foo1, w[i] does hold a character.

int main (int argc, char\*\*argv)

{

Int continue1=8; 1. Creating and initializing a variable (by hardcoding)

int num=4;

int num2=9;

int \*ptr=&num; 2 Creating and initializing a variable (by storing the result of an operation)

char\* words[]={"one", "two", "three", "four"}; 3 Creating and initializing a variable (by hardcoding)

while(continue1)

{

printf("\n\nEnter a number: 1, 2 or 3 to quit: \n");

scanf("%d", ptr); 4 Assigning a value to a variable (by storing the result of an operation)

if(num==1)

{

printf("Enter a number: ");

scanf("%d", ptr);

num2=foo(ptr); 5 Assigning a value to a variable (by storing a function return value/using a function)

printf("Number returned: %d\n", num2);

}

Else if(\*ptr==2)

{

num2=foo1(words); 6 Assigning a value to a variable (by storing a function return value/using a function)

printf("\nNumber returned:%d\n", num2);

}

Else if(\*ptr==3)

{

continue1=0; 7 Assigning a value to a variable (by hardcoding)

}

else

{

printf("Not a valid entry.\n");

}

}

}