DECLARATIONS AND INITIALIZATIONS

1. Answer & Explanation

Answer: Option **C**

Explanation:

fmod(x,y) - Calculates x modulo y, the remainder of x/y.
This function is the same as the modulus operator. But fmod() performs floating point divisions.

Example:

```
#include <stdio.h>
#include <math.h>

int main ()
{
  printf ("fmod of 3.14/2.1 is %lf\n", fmod (3.14,2.1) );
  return 0;
}
```

Output:
fmod of 3.14/2.1 is 1.040000

2. Answer & Explanation

Answer: Option **B**

Explanation:

External Linkage-> means global, non-static variables and functions.
Internal Linkage-> means static variables and functions with file scope.
None Linkage-> means Local variables.

3. Answer & Explanation

Answer: Option **D**

Explanation:

Variable names in C are made up of letters (upper and lower case) and digits. The underscore character ("_") is also permitted. Names must not begin with a digit.

Examples of valid (but not very descriptive) C variable names:
=> foo
=> Bar
=> BAZ
=> foo_bar
=> _foo42
=> _
=> QuUx

4. Answer & Explanation

Answer: Option **B**

Explanation:

extern int fun(); declaration in C is to indicate the existence of a global function and it is defined externally to the current module or in another file.

int fun(); declaration in C is to indicate the existence of a function inside the current module or in the same file.

5. Answer & Explanation

Answer: Option **A**

Explanation:

```
/* Example for ceil() and floor() functions: */

#include<stdio.h>
#include<math.h>

int main()
{
   printf("\n Result : %f" , ceil(1.44) );
   printf("\n Result : %f" , ceil(1.66) );

   printf("\n Result : %f" , floor(1.44) );
   printf("\n Result : %f" , floor(1.66) );
```

```
    return 0;
}
// Output:
// Result : 2.000000
// Result : 2.000000
// Result : 1.000000
// Result : 1.000000
```

6. Answer & Explanation

Answer: Option **B**

Explanation:

In computing, 'real number' often refers to non-complex floating-point numbers. It include both rational numbers, such as 42 and 3/4, and irrational numbers such as pi = 3.14159265...

When the accuracy of the floating point number is insufficient, we can use the doubleto define the number. The double is same as float but with longer precision and takes double space (8 bytes) than float.

To extend the precision further we can use long double which occupies 10 bytes of memory space.

7. Answer & Explanation

Answer: Option **B**

Explanation:

C data types classification are

1. Primary data types

1. int
2. char
3. float
4. double
5. void

2. Secondary data types (or) User-defined data type

1. Array
2. Pointer

3. Structure
4. Union
5. Enum

So, clearly long int l = 2.35; is not User-defined data type.
(i.e.long int l = 2.35; is the answer.)

8. Answer & Explanation

Answer: Option **A**

Explanation:

Declaring is the way a programmer tells the compiler to expect a particular type, be it a variable, class/struct/union type, a function type (prototype) or a particular object instance. (ie. extern int i)

Declaration never reserves any space for the variable or instance in the program's memory; it simply a "hint" to the compiler that a use of the variable or instance is expected in the program. This hinting is technically called "forward reference".

9. Answer & Explanation

Answer: Option **C**

Explanation:

extern int x; - is an external variable declaration.

double pow(double, double); - is a function prototype declaration.

Therefore, 1 and 3 are declarations. 2 is definition.

10. Answer & Explanation

Answer: Option **A**

Explanation:

- During declaration we tell the datatype of the Variable.

- During definition the value is initialized.

11. Answer & Explanation

Answer: Option **B**

Explanation:

A function prototype in C or C++ is a declaration of a function that omits the function body but does specify the function's name, argument types and return type.

While a function definition specifies what a function does, a function prototype can be thought of as specifying its interface.

www.vtucs.c

FLOATING POINT ISSUES

1. Answer & Explanation

Answer: Option **C**

Explanation:

The floating point data types are called real data types. Hence float, double, andlong double are real data types.

2. Answer & Explanation

Answer: Option **B**

Explanation:

Given 3.14 is a double constant.

To specify 3.14 as long double, we have to add L to the 3.14. (i.e 3.14L)

3. Answer & Explanation

Answer: Option **C**

4. Answer & Explanation

Answer: Option **A**

Explanation:

The range of long double is $3.4E^{-4932}$ to $1.1E^{+4932}$

5. Answer & Explanation

Answer: Option **B**

Explanation:

math.h is a header file in the standard library of C programming language designed for basic mathematical operations.

Declaration syntax: double log(double);

6. Answer & Explanation

Answer: Option **A**

Explanation:

Rounding off a value means replacing it by a nearest value that is approximately equal or smaller or greater to the given number.

y = (int)(x + 0.5); here x is any float value. To roundoff, we have to typecastthe value of x by using (int)

Example:

```
#include <stdio.h>

int main ()
{
  float x = 3.6;
  int y = (int)(x + 0.5);
  printf ("Result = %d\n", y );
  return 0;
}
```

Output:
Result = 4.

7. Answer & Explanation

Answer: Option **B**

8. Answer & Explanation

Answer: Option **D**

9. Answer & Explanation

Answer: Option **B**

10. Answer & Explanation

Answer: Option **C**

Explanation:

fmod(x,y) - Calculates x modulo y, the remainder of x/y.
This function is the same as the modulus operator. But fmod() performs floating point divisions.

Example:

```
#include <stdio.h>
#include <math.h>

int main ()
{
  printf ("fmod of 5.5 by 1.3 is %lf\n", fmod (5.5, 1.3) );
  return 0;
}
```

Output:
fmod of 5.5 by 1.3 is 0.300000

POINTERS

1. Answer & Explanation

Answer: Option **A**

2. Answer & Explanation

Answer: Option **C**

3. Answer & Explanation

Answer: Option **C**

Explanation:

The macro "NULL" is defined in locale.h, stddef.h, stdio.h, stdlib.h, string.h, time.h, and wchar.h.

4. Answer & Explanation

Answer: Option **A**

Explanation:

near=2, far=4 and huge=4 pointers exist only under DOS. Under windows and Linux every pointers is 4 bytes long.

5. Answer & Explanation

Answer: Option **D**

6. Answer & Explanation

Answer: Option **B**

7. <u>Answer & Explanation</u>

Answer: Option **C**

8. Answer & Explanation

Answer: Option **A**

www.vtucs.com

LIBRARY FUNCTIONS

1. Answer & Explanation

Answer: Option **D**

Explanation:

rewind() takes the file pointer to the beginning of the file. so that the next I/O operation will take place at the beginning of the file.
Example: rewind(FilePointer);

2. Answer & Explanation

Answer: Option **C**

Explanation:

stdio.h, which stands for "standard input/output header", is the header in the C standard library that contains macro definitions, constants, and declarations of functions and types used for various standard input and output operations.

3. Answer & Explanation

Answer: Option **D**

Explanation:

strrchr() returns a pointer to the last occurrence of character in a string.

Example:

```
#include <stdio.h>
#include <string.h>

int main()
{
   char str[30] = "12345678910111213";
   printf("The last position of '2' is %d.\n",
          strrchr(str, '2') - str);
   return 0;
```

```
}
```

Output: The last position of '2' is 14.

4. Answer & Explanation

Answer: Option **C**

Explanation:

The standard error(stderr) stream is the default destination for error messages and other diagnostic warnings. Like stdout, it is usually also directed to the output device of the standard console (generally, the screen).

5. Answer & Explanation

Answer: Option **A**

Explanation:

1. itoa() converts an integer to a string.
2. ltoa() converts a long to a string.
3. ultoa() converts an unsigned long to a string.
4. sprintf() sends formatted output to a string, so it can be used to convert any type of values to string type.

```
#include<stdio.h>
#include<stdlib.h>

int main(void)
{
  int   num1 = 12345;
  float num2 = 5.12;
  char str1[20];
  char str2[20];

  itoa(num1, str1, 10); /* 10 radix value */
  printf("integer = %d string = %s \n", num1, str1);

  sprintf(str2, "%f", num2);
  printf("float = %f string = %s", num2, str2);

  return 0;
```

```
}

// Output:
// integer = 12345 string = 12345
// float = 5.120000 string = 5.120000
```

6. Answer & Explanation

Answer: Option **A**

Explanation:

"fflush()" flush any buffered output associated with filename, which is either a file opened for writing or a shell command for redirecting output to a pipe or coprocess.

Example:
fflush(FilePointer);
fflush(NULL); flushes all streams.

7. Answer & Explanation

Answer: Option **A**

Explanation:

Do like this fprintf(stdout, "%s %d %f", str, i, a);

8. Answer & Explanation

Answer: Option **C**

Explanation:

The randomize() function initializes the random number generator with a random value based on time. You can try the sample program given below in Turbo-C, it may not work as expected in other compilers.

```
/* Prints a random number in the range 0 to 99 */
```

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

int main(void)
{
   randomize();
   printf("Random number in the 0-99 range: %d\n", random (100));
   return 0;
}
```

www.vtucs.com

INPUT / OUTPUT

1. Answer & Explanation

Answer: Option **C**

Explanation:

Declaration: char *fgets(char *s, int n, FILE *stream);

fgets reads characters from stream into the string s. It stops when it reads either n - 1 characters or a newline character, whichever comes first.

Therefore, the string str contain "I am a boy\n\0"

2. Answer & Explanation

Answer: Option **A**

Explanation:

The file source.txt will be opened in the binary mode.

3. Answer & Explanation

Answer: Option **B**

Explanation:

The fp is a structure which contains a char pointer which points to the first character of a file.

4. Answer & Explanation

Answer: Option **D**

Explanation:

r+ Open an existing file for update (reading and writing).

5. Answer & Explanation

Answer: Option **A**

Explanation:

To print a float value, %f is used as format specifier.

To print a double value, %lf is used as format specifier.

Therefore, the answer is printf("%f %lf", a, b);

6. Answer & Explanation

Answer: Option **D**

Explanation:

Extra parameter in call to fclose().

7. Answer & Explanation

Answer: Option **B**

Explanation:

The file source.txt is opened in read mode and target.txt is opened in write mode. The file source.txt contains "To err is human".

Inside the while loop,

ch=getc(fs); The first character('T') of the source.txt is stored in variable ch and it's checked for EOF.

if(ch==EOF) If EOF(End of file) is true, the loop breaks and program execution stops.

If not EOF encountered, fseek(fs, 4L, SEEK_CUR); the file pointer advances 4 character from the current position. Hence the file pointer is in 5th character of filesource.txt.

fputc(ch, ft); It writes the character 'T' stored in variable ch to target.txt.

The while loop runs three times and it write the character 1st and 5th and 11th characters ("Trh") in the target.txt file.

8. Answer & Explanation

Answer: Option **D**

Explanation:

To scan a float value, %f is used as format specifier.

To scan a double value, %lf is used as format specifier.

Therefore, the answer is scanf("%f %lf", &a, &b);

9. Answer & Explanation

Answer: Option **B**

Explanation:

Because, In fgets() we can specify the size of the buffer into which the string supplied will be stored.

10. Answer & Explanation

Answer: Option **B**

Explanation:

fp = fopen("DUMMY.C", "w"); A file DUMMY.C is opened in write mode and returns the file pointer to fp

t = fileno(fp); returns the handle for the fp stream and it stored in the variable t

printf("%d\n", t); It prints the handle number.

EXPRESSIONS

1. Answer & Explanation

Answer: Option **A**

Explanation:

C uses left associativity for evaluating expressions to break a tie between two operators having same precedence.

2. Answer & Explanation

Answer: Option **D**

Explanation:

Simply called as BODMAS (Bracket of Division, Multiplication, Addition and Subtraction).

How Do I Remember ? BODMAS !

- **B** - Brackets first
- **O** - Orders (ie Powers and Square Roots, etc.)
- **DM** - Division and Multiplication (left-to-right)
- **AS** - Addition and Subtraction (left-to-right)

3. Answer & Explanation

Answer: Option **C**

Explanation:

Option A: assignment statements are always return in paranthesis in the case of conditional operator. It should be a>b? (c=30):(c=40);

Option B: it is syntatically wrong.

Option D: syntatically wrong, it should be return(a>b ? a:b);

Option C: it uses nested conditional operator, this is logic for finding greatest number out of three numbers.

4. Answer & Explanation

Answer: Option **C**

Explanation:

Here, Multiplication will happen before the addition, but in which order the functions would be called is undefined. In an arithmetic expression the parenthesis tell the compiler which operands go with which operators but do not force the compiler to evaluate everything within the parenthesis first.

5. Answer & Explanation

Answer: Option **D**

Explanation:

An operation with only one operand is called unary operation.
Unary operators:
! Logical NOT operator.
~ bitwise NOT operator.
sizeof Size-of operator.

&& Logical AND is a logical operator.

Therefore, 1, 2, 3 are unary operators.

6. Answer & Explanation

Answer: Option **A**

Explanation:

2. Arithmetic operators: *, /, %, +, -
1. Relational operators: >, <, >=, <=, ==, !=
3. Logical operators : !, &&, ||
4. Assignment operators: =

STRINGS

1. Answer & Explanation

Answer: Option **B**

Explanation:

Declaration:

char *strnset(char *s, int ch, size_t n); Sets the first n characters of s toch

```
#include <stdio.h>
#include <string.h>

int main(void)
{
   char *string = "abcdefghijklmnopqrstuvwxyz";
   char letter = 'x';

   printf("string before strnset: %s\n", string);
   strnset(string, letter, 13);
   printf("string after  strnset: %s\n", string);

   return 0;
}
```

Output:

string before strnset: abcdefghijklmnopqrstuvwxyz

string after strnset: xxxxxxxxxxxxxnopqrstuvwxyz

2. Answer & Explanation

Answer: Option **C**

Explanation:

Declaration: strcmp(const char *s1, const char*s2);

The strcmp return an int value that is

if s1 < s2 returns a value < 0

if s1 == s2 returns 0

if s1 > s2 returns a value > 0

3. Answer & Explanation

Answer: Option **D**

Explanation:

The statement printf("\\n"); prints '\n' on the screen.

4. Answer & Explanation

Answer: Option **C**

Explanation:

Declaration: char *strrchr(const char *s, int c);

It scans a string s in the reverse direction, looking for a specific character c.

Example:

```
#include <string.h>
#include <stdio.h>

int main(void)
{
  char text[] = "I learn through IndiaBIX.com";
  char *ptr, c = 'i';

  ptr = strrchr(text, c);
  if (ptr)
    printf("The position of '%c' is: %d\n", c, ptr-text);
  else
    printf("The character was not found\n");
  return 0;
}
```

Output:

The position of 'i' is: 19

5. Answer & Explanation

Answer: Option **C**

Explanation:

The function strstr() Finds the first occurrence of a substring in another string

Declaration: char *strstr(const char *s1, const char *s2);

Return Value:
On success, strstr returns a pointer to the element in s1 where s2 begins (points tos2 in s1).
On error (if s2 does not occur in s1), strstr returns null.

Example:

```
#include <stdio.h>
#include <string.h>

int main(void)
{
   char *str1 = "IndiaBIX", *str2 = "ia", *ptr;

   ptr = strstr(str1, str2);
   printf("The substring is: %s\n", ptr);
   return 0;
}
```

Output: The substring is: iaBIX

6. Answer & Explanation

Answer: Option **C**

Explanation:

gets(); collects a string of characters terminated by a new line from the standard input stream stdin

```
#include <stdio.h>

int main(void)
{
   char string[80];

   printf("Enter a string:");
   gets(string);
   printf("The string input was: %s\n", string);
   return 0;
}
```

Output:

Enter a string: IndiaBIX

The string input was: IndiaBIX

7. Answer & Explanation

Answer: Option **A**

Explanation:

Option A is the correct function to find the length of given string.

Example:

```
#include<stdio.h>

int xstrlen(char *s)
{
    int length=0;
    while(*s!='\0')
    { length++; s++; }
    return (length);
}

int main()
```

```
{
    char d[] = "IndiaBIX";
    printf("Length = %d\n", xstrlen(d));
    return 0;
}
```

Output: Length = 8

www.vtucs.com

CONST

1. Answer & Explanation

Answer: Option **A**

Explanation:

Step 1: int y=128; The variable 'y' is declared as an integer type and initialized to value "128".

Step 2: const int x=y; The constant variable 'x' is declared as an integer and it is initialized with the variable 'y' value.

Step 3: printf("%d\n", x); It prints the value of variable 'x'.

Hence the output of the program is "128"

2. Answer & Explanation

Answer: Option **D**

Explanation:

The output will be (in 16-bit platform DOS):

K 75 0.000000

3. Answer & Explanation

Answer: Option **C**

4. Answer & Explanation

Answer: Option **C**

Explanation:

Step 1: const int x=5; The constant variable x is declared as an integer data type and initialized with value '5'.

Step 2: const int *ptrx; The constant variable ptrx is declared as an integer pointer.

Step 3: ptrx = &x; The address of the constant variable x is assigned to integer pointer variable ptrx.

Step 4: *ptrx = 10; Here we are indirectly trying to change the value of the constant vaiable x. This will result in an error.

To change the value of const variable x we have to use *(int *)&x = 10;

5. Answer & Explanation

Answer: Option **B**

6. Answer & Explanation

Answer: Option **C**

Explanation:

Step 1: const char *s = ""; The constant variable s is declared as an pointer to an array of characters type and initialized with an empty string.

Step 2: char str[] = "Hello"; The variable str is declared as an array of charactrers type and initialized with a string "Hello".

Step 3: s = str; The value of the variable str is assigned to the variable s. Thereforestr contains the text "Hello".

Step 4: while(*s){ printf("%c", *s++); } Here the while loop got executed untill the value of the variable s is available and it prints the each character of the variable s.

Hence the output of the program is "Hello"

7. Answer & Explanation

Answer: Option **C**

Explanation:

Step 1: int get(); This is the function prototype for the funtion get(), it tells the compiler returns an integer value and accept no parameters.

Step 2: const int x = get(); The constant variable x is declared as an integer data type and initialized with the value "20".

The function get() returns the value "20".

Step 3: printf("%d", x); It prints the value of the variable x.

Hence the output of the program is "20".

8. Answer & Explanation

Answer: Option **A**

Explanation:

Step 1: const int arr[5] = {1, 2, 3, 4, 5}; The constant variable arr is declared as an integer array and initialized to

arr[0] = 1, arr[1] = 2, arr[2] = 3, arr[3] = 4, arr[4] = 5

Step 2: printf("Before modification arr[3] = %d", arr[3]); It prints the value of arr[3] (ie. 4).

Step 3: fun(&arr[3]); The memory location of the arr[3] is passed to fun() andarr[3] value is modified to 10.

A const variable can be indirectly modified by a pointer.

Step 4: printf("After modification arr[3] = %d", arr[3]); It prints the value of arr[3] (ie. 10).

Hence the output of the program is

Before modification arr[3] = 4

After modification arr[3] = 10

9. Answer & Explanation

Answer: Option **D**

Explanation:

This program will show an error "Cannot modify a const object".

Step 1: const int i=0; The constant variable 'i' is declared as an integer and initialized with value of '0'(zero).

Step 2: printf("%d\n", i++); Here the variable 'i' is increemented by 1(one). This will create an error "Cannot modify a const object".

Because, we cannot modify a const variable.

10. Answer & Explanation

Answer: Option **B**

Explanation:

Step 1: const c = -11; The constant variable 'c' is declared and initialized to value "-11".

Step 2: const int d = 34; The constant variable 'd' is declared as an integer and initialized to value '34'.

Step 3: printf("%d, %d\n", c, d); The value of the variable 'c' and 'd' are printed.

Hence the output of the program is -11, 34

www.vu

COMPLICATED DECLARATIONS

1. **Answer & Explanation**

Answer: Option **B**

2. **Answer & Explanation**

Answer: Option **B**

3. **Answer & Explanation**

Answer: Option **D**

4. **Answer & Explanation**

Answer: Option **A**

5. **Answer & Explanation**

Answer: Option **C**

6. **Answer & Explanation**

Answer: Option **C**

7. **Answer & Explanation**

Answer: Option **C**

8. **Answer & Explanation**

Answer: Option **D**

9. Answer & Explanation

Answer: Option **B**

10. Answer & Explanation

Answer: Option **D**

11. Answer & Explanation

Answer: Option **B**

12. Answer & Explanation

Answer: Option **C**

www.vtucs.org