Technical Papers C + + Questions: C++ Questions Part I

Examrace Placement Series prepares you for the toughest placement exams to top companies.

1. Base class has some virtual method and derived class has a method with the same name. If we initialize the base class pointer with derived object, Calling of that virtual method will result in which method being called?

    a. Base method

    b. Derived method.

    Ans. b

2. For the following C program #define AREA (x) (3.14 * x * x) main () { float r1 = 6.25, r2 = 2.5, a; a = AREA (r1); printf (“\n Area of the circle is %f” a); a = AREA (r2); printf (“\n Area of the circle is %f” a); } What is the output? Ans. Area of the circle is 122.656250 Area of the circle is 19.625000

3. What do the following statements indicate. Explain. Int (* p) [10] int * f () int (* pf) () int * p[10] Refer to: Kernighan & Ritchie page no. 12 and Schaum series page no. 323

4. void main () { int d = 5; printf (“%f” d); } Ans: Undefined

5. void main () { int i; for (i = 1; i<4, i + +) switch (i) case 1: Printf (“%d” i); break; { case 2: Printf (“%d” i); break; case 3: Printf (“%d” i); break; } switch (i) case 4: Printf (“%d” i); } Ans: 1, 2, 3, 4

6. void main () { char * s = “\12345s\n” printf (“%d” sizeof (s)); } Ans: 6

7. void main () { unsigned i = 1;/* unsigned char k = -1 ⇒ k = 255; */signed j = -1;/* char k = -1 ⇒ k = 65535 *//* unsigned or signed int k = -1 ⇒ k = 65535 */if (iprintf (“less”); else if (i>j) printf (“greater”);) else if (i = = j) printf (“equal”); } Ans: Less

8. void main () { float j; j = 1000 * 1000; printf (“%f” j); }

    a. 1000000

    b. Overflow

    c. Error

    d. None

    Ans: 4

9. How do you declare an array of N pointers to functions returning pointers to functions returning pointers to characters? Ans: The first part of this question can be answered in at least three ways:

    a. char * (* (* a[N]) ()) ()

    b. Build the declaration up incrementally, using typedefs: Typedef char * pc;/* pointer to char */typedef pc fpc ();/* function returning pointer to char */typedef fpc * pfpc;/* pointer to above */typedef pfpc fpfpc ();/* function returning... */typedef fpfpc * pfpfpc;/* pointer to... */pfpfpc a[N];/* array of... */

    c. Use the cdecl program, which turns English into C and vice

    versa:

    cdecl> declare a as array of pointer to function returning

    pointer to function returning pointer to char

    char * (* (* a[]) ()) ()

    cdecl can also explain complicated declarations, help with

    casts, and indicate which set of parentheses the arguments

    go in (for complicated function definitions, like the one above).

    Any good book on C should explain how to read these complicated

    C declarations “inside out” to understand them (“declaration mimics use”).

    The pointer-to-function declarations in the examples above have

    not included parameter type information. When the parameters

    have complicated types, declarations can * really * get messy.

    (Modern versions of cdecl can help here, too.)

10. A structure pointer is defined of the type time. With 3 fields min, sec hours having pointers to intergers. Write the way to initialize the 2nd element to 10.

11. In the above question an array of pointers is declared. Write the statement to initialize the 3rd element of the 2 element to 10

12. int f () void main () { f (1); f (1, 2); f (1, 2, 3); } f (int i, int j, int k) { printf (“%d %d %d” i, j, k); } What are the number of syntax errors in the above? Ans: None.

13. void main () { int i = 7; printf (“%d” i + + * i + +); } Ans: 56

14. #define one 0 #ifdef one printf (“one is defined”); #ifndef one printf (“one is not defined”); Ans: “one is defined”

15. There was question in c working only on unix machine with pattern matching.

16. main () { static i = 3; printf (“%d” i--); return i>0? main (): 0; } Ans: 321

17. char * foo () { char result[100]; strcpy (result, “anything is good”); return (result); } void main () { char * j; j = foo () printf (“%s” j); } Ans: Anything is good.

18. void main () { char * s[] = { “dharma” “hewlett-packard” “siemens” “ibm” }; char * * p; p = s; printf (“%s” + + * p); rintf (“%s” * p + +); printf (“%s” + + * p); } Ans: “harma” (p → add (dharma) && (* p) → harma) “harma” (after printing, p → add (hewlett-packard) && (* p) → harma) “ewlett-packard”

19. Output of the following program is main () { int i = 0; for (i = 0; i<20; i + +) { switch (i) case 0: i + = 5; case 1: i + = 2; case 5: i + = 5; default i + = 4; break; } printf (“%d,” i); }

    a. 0, 5, 9, 13, 17

    b. 5, 9, 13, 17

    c. 12, 17, 22

    d. 16, 21

    e. Syntax error

    Answer: d

20. what is alloca ()

Ans: It allocates and frees memory after use/after getting out of scope