Enum Questions

1. What is Enum?

2. Give any two applications of enum?

3. Does,Enum follows scope?

4. Is it possible to change the enumerators?

5.predict the output:

#include<stdio.h>

enum tag{

a=10

{;

int main(void)

{

a=33;

return 0;

}

6. what are the ways to define a enumerator?

7. Explain the default values in enum?

8.what are the differnet ways to assign value for the enumertors?

9.predict the output:

#include<stdio.h>

enum tag{

a=(int)10.33

{;

int main(void)

{

printf(“%d\n”,a);

return 0;

}

10. predict the output:

#include<stdio.h>

float var = 10.33;

enum tag{

a=(int)var

{;

int main(void)

{

printf(“%d\n”,a);

return 0;

}

Calculate SIZEOF all the structures and print their address,

and reason out why this so.

1 struct node {

int a;

char c;

int b;

};

2. struct link {

char a;

short i;

};

3.struct pan {

char ch;

int a;

char p;

};

4. struct can {

char m;

char c;

int n;

};

5.struct mms {

char ch;

short s;

int n;

};

6.struct nm {

char c;

};

7.struct art {

short s;

shortv n;

char ch;

};

8.struct quier {

short s;

short n;

int m;

};

9. struct covet {

short s;

int n;

short m;

}

**Pointers :**

1.int main ()

{

int num;

char \*str = "Global edge";

printf ("0x%08x\n","Global edge");

printf ("%s\n","Global edge");

num = printf ("%d\n","Global edge");

num = printf ("%s\n","Globaledge");

printf ("%d\n", num);

return 0;

}

2. void fun (char \*s)

{

printf ("0x%08x", s);

return ;

}

int main ()

{

fun("Some people are not worth for trust");

return 0;

}

3. int main ()

{

char amsg[] = "Global";

char \*pmsg = "edge";

printf ("%p\n",amsg);

printf ("%p\n",pmsg);

printf ("%s\n",amsg);

printf ("%p\n",\*pmsg);

printf ("%s\n", \*pmsg);

return 0;

}

4. int main ()

{

char msg[] = "hi";

char dmsg[3];

// dmsg = msg;

char \*pmsg = "Hello";

char \*pdmsg;

pdmsg = pmsg;

printf ("%p\n", pmsg);

printf ("%p\n", pdmsg);

printf ("%s\n", pmsg);

printf ("%p\n", msg);

return 0;

}

6. char \*n;

int main ()

{

char \*p = "Higlobal";

char s[8] = "Hello";

// p = &s[0];

printf ("%p\n",p);

printf ("%p\n",&s);

\*p = 'p';

printf ("%d\n",\*p);

p = p + 8;

printf ("%p\n",p);

\*p = 0;

printf ("%p\n",n);

p = n;

\*p = n;

printf ("%s\n", s[0]);

return 0;

}

7. int main ()

{

int a = 10;

int b = &a;

printf ("1 = %c\n", b);

printf ("2 = %08x\n", b);

printf ("3 = %p\n", b);

return 0;

}

8 .int main()

{

int \*p = 10;

p++;

printf ("p = %08x\n", p);

return 0;

}

9. int main ()

{

char \*a[4] = {"global","edge","soft","limited"};

printf ("1= %s\n",\*(a+1));

// printf ("2 = %s\n",\*\*a);

printf ("3 = %s\n",a+1);

return 0;

}

10.int main ()

{

int \*p;

p = 100;

\*p = 19;

printf ("%d\n", \*p);

return 0;

}

11.int main()

{

int \*p = 10;

p++;

printf ("p = %d\n", p);

return 0;

}

12.void fun(int x)

{

x = 30;

}

int main()

{

int y = 20;

fun(y);

printf("%d", y);

return 0;

}

13.void fun(int \*ptr)

{

\*ptr = 30;

}

int main()

{

int y = 20;

fun(&y);

printf("%d", y);

return 0;

}

14.int main()

{

int arri[] = {1, 2 ,3};

int \*ptri = arri;

char arrc[] = {1, 2 ,3};

char \*ptrc = arrc;

printf("sizeof arri[] = %d ", sizeof(arri));

printf("sizeof ptri = %d ", sizeof(ptri));

printf("sizeof arrc[] = %d ", sizeof(arrc));

printf("sizeof ptrc = %d ", sizeof(ptrc));

return 0;

}

15.int main()

{

float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};

float \*ptr1 = &arr[0];

float \*ptr2 = ptr1 + 3;

printf("%f ", \*ptr2);

printf("%d", ptr2 - ptr1);

return 0;

}

16.int main()

{

int arr[] = {10, 20, 30, 40, 50, 60};

int \*ptr1 = arr;

char \*s = NULL;

int \*ptr2 = arr + 5;

printf("Number of elements between two pointer are: %d.",

(ptr2 - ptr1));

printf("Number of bytes between two pointers are: %d",

(char\*)ptr2 - (char\*) ptr1);

printf("%d\n", sizeof(s));

return 0;

}

17.int main()

{

char \*ptr = "Globaledge";

printf("%c\n", \*&\*&\*ptr);

return 0;

}

18.void fun(int arr[])

{

int i;

int arr\_size = sizeof(arr)/sizeof(arr[0]);

for (i = 0; i < arr\_size; i++)

printf("%d ", arr[i]);

}

int main()

{

int i;

int arr[] = {10, 20 ,30, 40};

fun(&arr);

return 0;

}

19.int main()

{

int a = 12;

void \*ptr = (int \*)&a;

printf("%d", \*ptr);

return 0;

}

20.int main()

{

int arr[] = {1, 2, 3, 4, 5};

int \*p = arr;

++\*p;

p += 2;

printf("%d", \*p);

return 0;

}

21.int main()

{

int var; /\*Suppose address of var is 2000 \*/

\*ptr = &var;

void \*ptr = &var;

\*ptr = 5;

printf("var=%d and \*ptr=%d",var,\*ptr);

return 0;

}

22.int main()

{

int i;

int arr[] = {10, 20 ,30, 40};

int arr\_size = sizeof(arr) / sizeof(arr[0]);

for (i = 0; i < arr\_size; i++)

printf("%d ", arr[i]);

return 0;

}

23.int main()

{

int a[4] = {1,2};

printf("%p\n", &a);

return 0;

}

24.int main()

{

int main = 3;

printf("%s\n", printf());

return 0;

}

25.int main ()

{

int x = 65, \*p = &x;

void \*q=p;

char \*r=q;

printf("%c",\*r);

return 0;

}

26.void f() {

printf ("Hello\n");

}

int main ()

{

;

// return 0;

}

27.void swap(int m, int n)

{

int x = m;

m = n;

n = x;

}

main()

{

int x=5, y=3;

swap(x,y);

printf("%d %d", x, y);

}

28.int main()

{

int x = 1;

switch(x)

{

default: printf("Hello");

case 1: printf("hi"); break;

}

}

29.main()

{

int a[] = {1,2}, \*p = a;

printf("%d", p[1]);

}

30.int main ()

{

char a[15] = "GlobalEdge";

a[3] = 'p';

printf ("%s\n",a);

return 0;

}

31.int main ()

{

char \*a = "GlobalEdge";

\*(a+3) = 'p';

printf ("%s\n",a);

return 0;

}

32.int main ()

{

char \*s = "globaledge";

printf ("%lu",sizeof(s));

return 0;

}

33.int main ()

{

int a = 5;

if (a=5){

while(a=5)

break;

printf ("hi\n");

}

printf ("hello\n");

return 0;

}

34.int main ()

{

int i = 10;

switch (i)

{

int i = 20;

case 10 : printf ("%d\n",i);

break;

}

return 0;

}

35.int main ()

{

int i = 10;

switch (i)

{

// int i = 20;

case 10 :

{// int i = 30;

printf ("%d\n",i);

// int p = 20;

printf ("%d\n",i);

// break;

}

printf ("%d\n",i);

}

return 0;

}

36.int main ()

{

int \*p;

int \*s;

p = malloc(20);

s = p;

free(p);

p = NULL;

// free(s);

// s = NULL;

return 0;

}

**Data types and Declaration ,defination**

1.int main()

{

unsigned int x = -1;

int y = ~0;

if (x == y)

printf("same");

else

printf("not same");

return 0;

}

2.int main()

{

char a = '\012';

printf("%d", a);

return 0;

}

3.int main()

{

char c = 125;

c = c+10;

printf("%d", c);

return 0;

}

4.int main()

{

if (sizeof(int) > -1)

printf("Yes");

else

printf("No");

return 0;

}

5.int main()

{

float x = 0.1;

if ( x == 0.1 )

printf("IF");

else if (x == 0.1f)

printf("ELSE IF");

else

printf("ELSE");

}

6.int var = 20;

int main()

{

int var = var;

printf("%d ", var);

return 0;

}

7.extern int var;

int main()

{

var = 10;

printf("%d ", var);

return 0;

}

8.extern int var = 0;

int main()

{

var = 10;

printf("%d ", var);

return 0;

}

9.int main()

{

{

int var = 10;

}

{

printf("%d", var);

}

return 0;

}

10.int main()

{

int x = 1, y = 2, z = 3;

printf(" x = %d, y = %d, z = %d \n", x, y, z);

{

int x = 10;

float y = 20;

printf(" x = %d, y = %f, z = %d \n", x, y, z);

{

int z = 100;

printf(" x = %d, y = %f, z = %d \n", x, y, z);

}

}

return 0;

}

11.int main()

{

int x = 032;

printf("%d", x);

return 0;

}

**Storage class specifiers:**

1.int main()

{

static int i=5;

if(--i){

main();

printf("%d ",i);

}

}

2. int main()

{

static int i=5;

if (--i){

printf("%d ",i);

main();

}

}

3.int main()

{

typedef static int \*i;

int j;

i a = &j;

printf("%d", \*a);

return 0;

}

4.int main()

{

typedef int i;

i a = 0;

printf("%d", a);

return 0;

}

5. int fun()

{

static int num = 16;

return num--;

}

int main()

{

for(fun(); fun(); fun())

printf("%d ", fun());

return 0;

}

6. int main()

{

int x = 10;

static int y = x;

if(x == y)

printf("Equal");

else if(x > y)

printf("Greater");

else

printf("Less");

return 0;

}

7.int f(int n)

{

static int i = 1;

if (n >= 5)

return n;

n = n+i;

i++;

return f(n);

}

8. int main()

{

register int i = 10;

int \*ptr = &i;

printf("%d", \*ptr);

return 0;

}

9. int main()

{

extern int i;

printf("%d ", i);

{

int i = 10;

printf("%d ", i);

}

}

10.int fun(int n)

{

static int s = 0;

s = s + n;

return (s);

}

int main()

{

int i = 10, x;

while (i > 0)

{

x = fun(i);

i--;

}

printf ("%d ", x);

return 0;

}

11. char \*fun()

{

static char arr[1024];

return arr;

}

int main()

{

char \*str = "global edge";

strcpy(fun(), str);

str = fun();

strcpy(str, "global edge");

printf("%s", fun());

return 0;

}