




quieres trabajar
en Wuolah??

TE BUSCAMOS

Preguntas: 10

Respuestas
válidas: 

Puntuación: 





Nota: 

1

Elección única

¿Cómo cree que se calcularía más rápido la operación "a = b * c" suponiendo que el valor de c es 5?

Usuario Profesores

-  a) $a = b + (b \ll 2);$
-  b) $a = b * c;$
-  c) $a = b + b + b + b + b;$
-  d) $a = c * b;$

2



Elección única

Dado el siguiente código y suponiendo el vector v inicializado, ¿qué opción es verdadera?

```
for (int i = 0; i < 1000; ++i)
```

```
{  
    if ((v[i] % 3) == 0)  
        foo(v[i]);  
    else  
        switch((v[i] % 3))  
        {  
            case 1: foo(v[i] + 2); break;  
            case 2: foo(v[i] + 1); break;  
        }  
}
```

Usuario Profesores

-  a) los valores contenidos en v no afectan a la velocidad de ejecución
-  b) la ejecución finaliza antes si v contiene muchos múltiplos de 3

sin ánimo
de lucro,
chequea esto:



tú puedes
ayudarnos a
llevar
WUOLAH
al siguiente
nivel
(o alguien que
conozcas)

- ☒ c) la ejecución finaliza antes si v no contiene ningún múltiplo de 3
- ☒ d) sólo el desenrollado de bucle puede servir para optimizar el código

3

Elección única

¿Qué código cree que calculará de forma correcta y en menor tiempo el producto de dos matrices en un sistema multiprocesador? Suponga matrices cuadradas, c inicializada a cero y N muy grande.

`int a[N][N], b[N][N], c[N][N];`

Usuario Profesores

- ☒ a)

```
for (int i=0; i<N; ++i)
    #pragma omp parallel for
    for (int j=0; j<N; ++j)
        for (int k=0; k<N; ++k)
            c[i][j] += a[i][k] * b[k][j];
```
- ☒ b)

```
for (int i=0; i<N; ++i)
    #pragma omp parallel for
    for (int j=0; j<N; ++j)
        for (int k=0; k<N; ++k)
            #pragma omp atomic
            c[i][j] += a[i][k] * b[k][j];
```
- ☒ c)

```
for (int i=0; i<N; ++i)
    #pragma omp parallel for
    for (int j=0; j<N; ++j)
        for (int k=0; k<N; ++k)
            #pragma omp critical
            c[i][j] += a[i][k] * b[k][j];
```
- ☒ d)

```
for (int i=0; i<N; ++i)
    for (int j=0; j<N; ++j)
        for (int k=0; k<N; ++k)
            c[i][j] += a[i][k] * b[k][j];
```

4

Elección única

Indique qué opción se ejecutará más rápido dados

`const int n = 1000000;`

`int a[n], b[n];`

Usuario Profesores

- ☒ a)

```
for (i=0 ; i<n ; i++) {
    *p = *p + a[i]*b[i];
}
```
- ☒ b)

```
for (i=0 ; i<n ; ++i) {
    *p += a[i]*b[i];
}
```
- ☒ c)

```
int tmp0=0, tmp1=0, tmp2=0, tmp3=0;
for (i=0 ; i<n ; i+=4) {
    tmp0 += a[i ]*b[i ];
    tmp1 += a[i+1]*b[i+1];
    tmp2 += a[i+2]*b[i+2];
    tmp3 += a[i+3]*b[i+3];
}
*p = tmp0 + tmp1 + tmp2 + tmp3;
```
- ☒ d)

```
for (i=0 ; i<n ; i+=4) {
    *p += a[i ]*b[i ];
    *p += a[i+1]*b[i+1];
    *p += a[i+2]*b[i+2];
    *p += a[i+3]*b[i+3];
}
```

```

    *p := a[i+2]*b[i+2],
    *p += a[i+3]*b[i+3];
}

```

5

¿Cuál de las siguientes ventajas aporta la técnica de desenrollado de bucle?

Elección única

Usuario Profesores

- ☒ a) elimina todas las instrucciones de salto del bucle
- ☒ b) ninguna otra respuesta es correcta
- ☒ c) aumenta la velocidad de ejecución
- ☒ d) reduce el tamaño del código

6

¿Cuál de las siguientes formas de implementar el mismo algoritmo cree más rápida?

Elección única

```

const int N = 5000, REP = 40000;
int R[REP + 1];
struct S { int a, b; } s[N];

```

Usuario Profesores

- ☒ a)

```
for ( int ii = 1; ii <= REP ; ++ ii )
{
    int X1 = 0 , X2 = 0;
    for ( int i = 0; i < N ; ++ i )
        X1 += 2 * s [ i ]. a + ii;
    for ( int i = 0; i < N ; ++ i )
        X2 += 3 * s [ i ]. b - ii;
    if ( X1 < X2 )
        R [ ii ] = X1;
    else
        R [ ii ] = X2;
}
```
- ☒ b)

```
for ( int ii = 1; ii <= REP ; ++ ii )
{
    int x1 = 0 , x2 = 0;
    for ( int i = 0; i < N ; ++ i )
    {
        x1 += 2 * s [ i ]. a + ii;
        x2 += 3 * s [ i ]. b - ii;
    }
    R [ ii ] = std :: min ( x1 , x2 );
}
```
- ☒ c)

```
struct { int x1 , x2; } x[N];

for ( int i = 0; i < N ; ++ i )
{
    x [ i ]. x1 = 2 * s [ i ]. a ;
    x [ i ]. x2 = 3 * s [ i ]. b ;
}

for ( int ii = 1; ii <= REP ; ++ ii )
{
    int x1 = 0 , x2 = 0;
    for ( int i = 0; i < N ; ++ i )
    {
        x1 += x [ i ]. x1 + ii ;
        x2 += x [ i ]. x2 - ii ;
    }
    R [ ii ] = std :: min ( x1 , x2 ) ;
}
```

quieres trabajar
en Wuolah??

TE BUSCAMOS

sin ánimo
de lucro,
chequea esto:



tú puedes
ayudarnos a
llevar
WUOLAH
al siguiente
nivel
(o alguien que
conozcas)

7
Elección única

¿Cuál de los siguientes códigos dirías que tiene menor tiempo de ejecución?
Usuario Profesores

- ☐ d)

```
int sa = 0 , sb = 0;
for (int i = 0; i < N ; ++ i)
{
    sa += s [i]. a;
    sb += s [i]. b;
}
sa *= 2;
sb *= 3;
for (int ii = 1; ii <= REP ; ++ ii)
    R[ii] = std :: min (sa + N * ii , sb - N * ii);
```
- ☐ a) Todas las opciones tienen el mismo tiempo de ejecución.
- ☐ b)

```
static char *classes = "WSU";
letter = classes [queue];
```
- ☐ c)

```
if ( queue == 0)
    letter = 'W';
else if ( queue == 1)
    letter = 'S';
else
    letter = 'U';
```
- ☐ d)

```
switch ( queue ) {
case 0 : letter = 'W';
break;
case 1 : letter = 'S';
break;
case 2 : letter = 'U';
break;
}
```

8
Elección única

¿Cuál de las siguientes versiones de una función que multiplica un entero por 6 cree que se obtendrá al compilar con optimización en espacio (-Os)?

```
int f(int x)
{
    return x * 6;
}
```

Usuario Profesores

- ☐ a) ninguna otra respuesta es correcta
- ☐ b)

```
0x401116 <+0>: imul    $0x6,%edi,%eax
0x401119 <+3>: retq
```
- ☐ c)

```
0x401106 <+0>: lea    (%rdi,%rdi,2),%eax
0x401109 <+3>: add    %eax,%eax
0x40110b <+5>: retq
```
- ☐ d)

```
0x401106 <+0>: push   %rbp
0x401107 <+1>: mov    %rsp,%rbp
0x40110a <+4>: mov    %edi,-0x4(%rbp)
0x40110d <+7>: mov    -0x4(%rbp),%edx
0x401110 <+10>: mov    %edx,%eax
0x401112 <+12>: add    %eax,%eax
0x401114 <+14>: add    %edx,%eax
0x401116 <+16>: add    %eax,%eax
0x401118 <+18>: pop    %rbp
0x401119 <+19>: retq
```

WUOLAH

9

Elección única

¿Cuál es la principal diferencia entre las opciones de optimización -O2 y -O3?
Usuario Profesores

- ☐ a) -O3 solo funciona para procesadores Intel, mientras que -O2 puede usarse para cualquier tipo de procesadores
- ☐ b) -O3 crea un binario que tendrá una ejecución más eficiente que -O2, aunque ambos binarios pueden ser ejecutados en máquinas diferentes de las que los crearon
- ☐ c) -O3 crea un binario más pequeño que -O2 a costa de perder un poco de rendimiento
- ☒ d) -O2 crea un binario que puede ser usado en cualquier modelo de CPU, mientras que -O3 lo crea en función del que compila el código

10

Elección única

¿Qué forma de ordenar las opciones *case* que aparecen en una sentencia *switch* ofrece mejores prestaciones?
Usuario Profesores

- ☐ a) deben aparecer primero las opciones que tengan una mayor probabilidad de ser ciertas
- ☐ b) deben aparecer primero las opciones que contengan un menor número de instrucciones
- ☐ c) deben aparecer primero las opciones que contengan un mayor número de instrucciones
- ☒ d) el orden de aparición es indiferente