UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

INE018 MATEMÁTICA COMPUTACIONAL

Solucionario del examen parcial
2024-1

Indicaciones generales:

- Duración: 120 minutos.
- No está permitido el uso de ningún material o equipo electrónico adicional al indicado (no celulares, no tablets, no libros).
- **La presentación, la ortografía y la gramática de los trabajos influirán en la calificación.**

Puntaje total: 20 puntos.

Pregunta 1. (2 puntos)

¿Cuál es el nombre de la función que debe ser definida en todos los programas en C++? ¿Qué sentencia típicamente aparece al final de dicha función?

Todo programa en C++ debe contener una función `main`. Esta función especifica el punto inicial del cómputo. La última sentencia en `main` suele ser `return 0;` la cual por convención reporta el estado del programa indicando que el valor de la función es 0. Un valor de 0 significa éxito, otro valor es señal de que algo falló.

Pregunta 2. (2 puntos)

Indique los valores y tipos de las siguientes expresiones:

a. `2 + 3` es `int` y vale `5`.

b. `19.0 / 5` es `double` y vale `3.8`.

c. `19 % 5` es `int` y vale `4`.

d. `6 + 5 / 4 - 3` es `int` y vale `4`.

e. `10 + 9 * ((8 + 7) % 6) + 5 * 4 % 3 * 2 + 1` es `int` y vale `42`.

f. `8 + 5 * 3 / 2` es `int` y vale `15`.

g. `3 / 14 / 7 / (1.0 * 2) + 10 / 6` es `double` y vale `1.0`.

h. `1 / 2 > 0 || 4 == 9 % 5 || 1 + 1 < 1 - 1` es `bool` y vale `true`.

Pregunta 3. (2 puntos)

¿Qué línea de control de bucle `for` usaría para contar de siete en siete comenzando desde cero hasta que el número tenga más de dos dígitos?

```
for (int i = 0; i < 100; i += 7)
```

Pregunta 4. (2 puntos)

Las variables declaradas dentro de una función son llamadas *variables locales*. ¿Cuál es el significado de la palabra *local* en este contexto?

Local significa que el alcance de la variable se extiende hasta el final del bloque en el que está declarada.

Pregunta 5. (2 puntos)

Justifique la veracidad o falsedad del siguiente enunciado:
Si ejecutamos las líneas

```
string linea;
cin >> linea;
```

el programa leerá la línea entera de datos del usuario y la guardará en la variable `linea`.

Falso. El operador `>>` se detiene ni bien encuentra el primer whitespace. Por ejemplo, si ingreso la cadena "`Manuel Loaiza`", solo la cadena "`Manuel`" fue guardada en `linea`.

Pregunta 6. (2 puntos)

Dado el siguiente programa:

```
void Imprimir(int dos, int uno, int tres) {
    cout << tres << " es casi " << dos << " + " << uno << endl;
}

void Misterio(void) {
    int uno = 4;
    int dos = 3;
    int tres = 10;
    int n = 17;
    int cuatro = 3;
    Imprimir(uno, dos, tres);
    Imprimir(tres, cuatro, 5);
    Imprimir(2, dos * 2, n);
    Imprimir(n, tres * uno, cuatro);
    Imprimir(tres - cuatro, uno, dos);
}
```

Escriba la salida de cada una de las llamadas tal como aparecerían en la consola.

```
10 es casi 4 + 3
5 es casi 10 + 3
```

```
17 es casi 2 + 6
3 es casi 17 + 40
3 es casi 7 + 4
```

Pregunta 7. (2 puntos)

Para cada llamada a la función

```
int Misterio(int n) {
    if (n < 0) {
        n *= 3;
        return n;
    } else {
        n = n + 3;
    }
    if (n % 2 == 1) {
        n += n % 10;
    }
    return n;
}
```

escriba el valor retornado:

```
Misterio(-5);  // -15
Misterio(0);  // 6
Misterio(7);  // 10
Misterio(18);  // 22
Misterio(49);  // 52
```

Pregunta 8. (2 puntos)

Para cada llamada a la función

```
int Misterio(int i, int j) {
    int k = 0;
    while (i > j) {
        i = i - j;
        k = k + (i - 1);
    }
    return k;
}
```

escriba el valor retornado:

```
Misterio(2, 9);  // 0
Misterio(5, 1);  // 6
Misterio(38, 5);  // 119
Misterio(5, 9);  // 0
Misterio(40, 10);  // 57
```

Pregunta 9. (2 puntos)

Escriba un procedimiento llamado `ImprimirMultiplos` que acepte un entero n y un entero m como parámetros e imprima una línea reportando los m primeros múltiplos de n. Por ejemplo, las siguientes llamadas

```
ImprimirMultiplos(3, 5);
ImprimirMultiplos(7, 3);
```

deben producir la siguiente salida:

```
Los 5 primeros múltiplos de 3 son 3, 6, 9, 12 y 15.
Los 3 primeros múltiplos de 7 son 7, 14 y 21.
```

Note la separación entre los múltiplos y los signos de puntuación. Debe reproducir exactamente este formato. Asuma que el número de múltiplos solicitados es mayor que 1.

```
void ImprimirMultiplos(int n, int m) {
    cout << "Los " << m << " primeros múltiplos de " << n << " son";
    for (int i = 1; i <= m; i++) {
        if (i == m) {
            cout << " y";
        } else if (i > 1) {
            cout << ",";
        }
        cout << " " << i * n;
    }
    cout << "." << endl;
}
```

Pregunta 10. (2 puntos)

Escriba un procedimiento llamado `TresCaras` que repetitivamente lanza una moneda hasta obtener tres caras seguidas. Cada vez que la moneda es lanzada, se imprime lo obtenido (C para caras, S para sellos). Cuando se consiguen tres caras consecutivas, imprima un mensaje de felicitación. A continuación tiene una posible salida a una llamada a la función `TresCaras`:

```
S S S C S C C C
Tres caras consecutivas :)
```

Utilice la función `rand` de la biblioteca `<cstdlib>` para generar números pseudoaleatorios.

Observación: Si leemos la documentación de la función `rand`, encontramos que retorna un número pseudoaleatorio entre `0` y `RAND_MAX`. En la documentación de `RAND_MAX` encontramos que su valor depende de la implementación y solo nos garantiza que es al menos 32767. Desde C++11 existen mejores generadores de números pseudoaleatorios. En particular, la biblioteca `<random>` nos provee un Mersenne Twistter basado en el primo $2^{19937}-1$ llamado `mt19937`. Asimismo, en esta biblioteca encontramos distribuciones como

`uniform_int_distribution` la cual nos da perfectamente números uniformemente distribuidos sin el sesgo del residuo. Yo suelo utilizar lo mencionado de la siguiente manera:

```
int64_t seed = std::chrono::steady_clock::now().time_since_epoch().count();
std::mt19937_64 gen(seed);
long Random(long l, long r) {
    return std::uniform_int_distribution<long>(l, r)(gen);
}

void TresCaras(void) {
    int caras_consecutivas = 0;
    while (caras_consecutivas < 3) {
        if (rand() % 2 == 1) {
            cout << "C ";
            caras_consecutivas++;
        } else {
            cout << "S ";
            caras_consecutivas = 0;
        }
    }
    cout << endl;
    cout << "Tres caras consecutivas :)" << endl;
}
```

Profesor del curso: Manuel Loaiza Vasquez.

Lima, 6 de junio de 2024.