1 Makro: MAX

1.1 Aufgabe

Sie haben folgendes Makro mit dem Präprozessor definiert:

```
1 #define MAX(a, b) ((a) < (b) ? (b) : (a))
```

Und in `main()` folgende Anweisung geschrieben:

```
1 int x = 5, y = 10;
2 int z = MAX(x++, y++);
```

Welches Resultat erhalten Sie? Entspricht es dem, was Sie erwartet haben? Wenn nein, korrigieren Sie das Programmfragment.

1.2 Lösung

Resultat:

```
1 int z = ((x++) < (y++) ? (y++) : (x++)); // x = 6, y = 12, z = 11
```

Erwartet wird:

```
1 int z = ((6) < (11) ? (11) : (6)); // x = 6, y = 11, z =  11
```

Korrektur:

```
1 int x = 5, y = 10;
2 x++;
3 y++;
4 int z = MAX(x, y);
```

2 Speicherbedarf

2.1 Aufgabe

Messen Sie auf Ihrem System den Speicherbedarf der primitiven Datentypen und geben Sie die Resultate tabellarisch aus:

```
char        1 Byte
short       2 Bytes
...
```

Tipp: Die Funktion `sizeof()` könnte weiterhelfen.

Knacknuss: Verwenden Sie ein parametrisiertes Makro zur starken Vereinfachung des Programmcodes.

2.2 Lösung

```
#define TypeSize(type) setw(12) << left << #type << sizeof(type) << \
    (sizeof(type) == 1 ? " Byte" : " Bytes")

void typesizes() {
    std::cout << TypeSize(bool) << std::endl;
    std::cout << TypeSize(char) << std::endl;
    std::cout << TypeSize(short) << std::endl;
    std::cout << TypeSize(int) << std::endl;
    std::cout << TypeSize(long) << std::endl;
    std::cout << TypeSize(long long) << std::endl;
    std::cout << TypeSize(float) << std::endl;
    std::cout << TypeSize(double) << std::endl;
    std::cout << TypeSize(long double) << std::endl;
}
```