MODERN C++ DESIGN PATTERNS

Structure Bit-Fields

by Prasanna Ghali

Packing/Unpacking Members (1/6)

- How much memory required to store object of type `PlayerAttributes`?

```
struct PlayerAttributes {
  using UC = uint8_t;
  using US = uint16_t;

  // comments represent range of values
  UC level;  // 0 - 3
  UC power;  // 0 - 63
  US range;  // 0 - 1023
  UC armor;  // 0 - 15
  US health; // 0 - 511
  UC grade;  // 0 - 1
};
```

Packing/Unpacking Members (2/6)

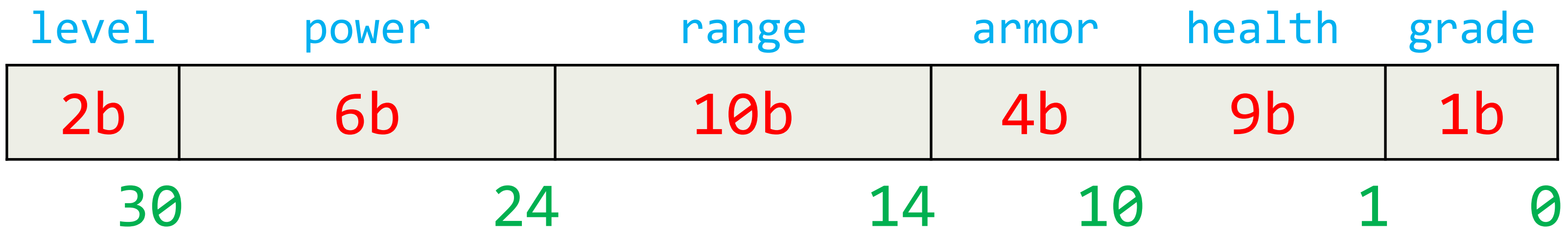
- Why use 10 bytes to store 32 bits of data?

```
struct PlayerAttributes {
  using UC = uint8_t;
  using US = uint16_t;

  // comments represent range of values
  UC level;  // 0 - 3
  UC power;  // 0 - 63
  US range;  // 0 - 1023
  UC armor;  // 0 - 15
  US health; // 0 - 511
  UC grade;  // 0 - 1
};
```

Packing/Unpacking Members (3/6)

- Why not pack values of structure members into a variable of type `uint32_t`?



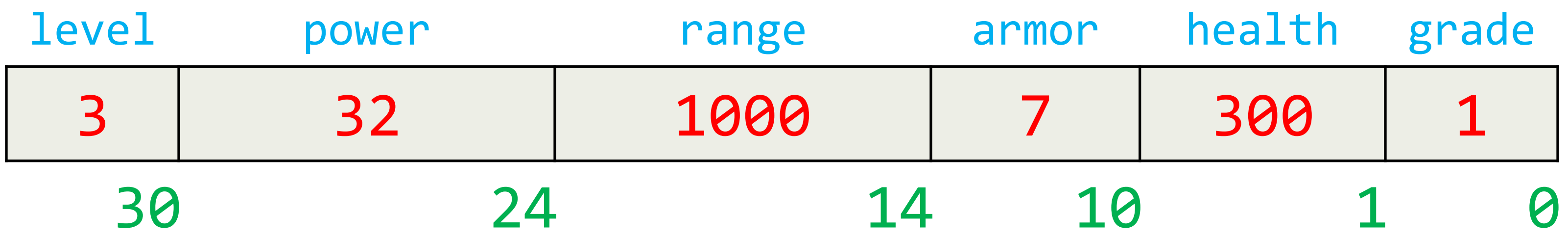
Packing/Unpacking Members (4/6)

- Need to set different bits of `uint32_t` variable to these values:

level	power	range	armor	health	grade
3	32	1000	7	300	1
30	24	14	10	1	0

Packing/Unpacking Members (5/6)

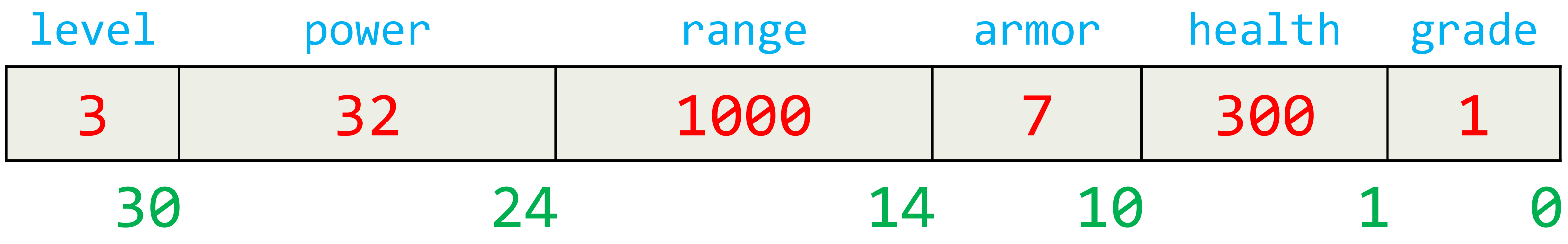
- Bit twiddling required to set different bits of `uint32_t` to these values:



```
uint32_t attrs;
attrs = 3U << 30;        // set level to 3
attrs |= 32U << 24;      // set power to 32
attrs |= 1000U << 14;    // set range to 1000
attrs |= 7U << 10;       // set armor to 7
attrs |= 300U << 1;      // set health to 300
attrs |= 1U;             // set grade to 1
```

Packing/Unpacking Members (6/6)

- Again bit twiddling required to extract each attribute value from `uint32_t` value:



```
uint32_t attrs = (3U << 30) | (32U << 24) | (1000U << 14) |
                 (7U << 10) | (300U << 1) | 1U;
// later, unpack attrs to extract individual attributes
uint8_t  level  = (attrs & 3U<<30) >> 30;
uint8_t  power  = (attrs & 63U<<24) >> 24;
uint16_t range  = (attrs & 1023U<<14) >> 14;
uint8_t  armor  = (attrs & 15U<<10) >> 10;
uint16_t health = (attrs & 511U<<1) >> 1;
uint8_t  grade  = attrs & 1U;
```

Bit-Fields

- Classes, structures, and unions can contain members smaller than 8 bits:

```
struct PlayerAttributes {
  using UI = uint32_t;

  UI level    : 2;  // 0 - 3
  UI power    : 6;  // 0 - 63
  UI range    : 10; // 0 - 1023
  UI armor    : 4;  // 0 - 15
  UI health   : 9;  // 0 - 511
  UI grade    : 1;  // 0 - 1
};
```

Using Bit-Fields

- A bit-field is accessed in much same way as regular member:

```
struct PlayerAttributes {
  using UI = uint32_t;

  UI level    : 2;  // 0 - 3
  UI power    : 6;  // 0 - 63
  UI range    : 10; // 0 - 1023
  UI armor    : 4;  // 0 - 15
  UI health   : 9;  // 0 - 511
  UI grade    : 1;  // 0 - 1
};
```

```
PlayerAttributes pa;
pa.level  = 3U;
pa.power  = 32U;
pa.range  = 1000U;
pa.armor  = 7U;
pa.health = 300U;
pa.grade  = 1U;
++pa.grade;
```