Macros

Macros

- Constant Macros
- Flag Macros
- Predefined macros
- Function-like Macros
- Statement macros
- Related Operators

Constant Macros

- Example

```
#define PI      3.14159
  . . .
circumference = 2 * PI * radius;
```

- Appropriate use of parentheses required

```
/* This is a bad example */
#define REC1_SZ    10
#define REC2_SZ    5
#define TOT_REC_SZ REC1_SZ + REC2_SZ
  . . .
int     num_recs = 20,
  rec_bytes = num_recs * TOT_REC_SZ;
```

Constant Macros

- Style suggestion

  Enclose macro values in parentheses whenever possible, even if it's not strictly required

```
#define REC1_SZ      (10)
#define REC2_SZ      (5)
#define TOT_REC_SZ   (REC1_SZ + REC2_SZ)
```

Flag macros

```
#define DEBUG
  . . .
  strncpy( result, source, 20 );
  result[20] = NULL_CHAR;
#ifdef DEBUG
  printf( "Check 9: %s\n", result );
#endif
```

Predefined macros

```
printf( "Error detected in %s at %d\n",
  __FILE__,
  __LINE__
        );

__DATE__
"Oct 24 1994"

__TIME__
"09:47:23"

__STDC__
```

Function-like Macros

- Example
```
#define PI                  (3.14159)
#define AREA( r )     ( (r) * (r) * PI )
. . .
double    radius  = 8.5,
          surface = 0.0;
surface = AREA( radius );
```
 -- expands to --
```
surface = (radius) * (radius) * 3.14159;
```
- Important: there must be <u>NO</u> <u>SPACE</u> between the macro name, and the opening parenthesis

Function-like Macros

- Enclose macro arguments in parenthesis as required
```
/* This is a bad example */
#define AREA(r) ( r * r * PI )
. . .
double     radius  = 8.5,
           surface = 0.0;
surface = AREA( radius + 0.5 );
```
 -- expands to --
```
surface = ( radius + 0.5 * radius + 0.5 * 3.14159);
```
- Style suggestion

  Enclose macro arguments in parentheses whenever possible, even if it's not strictly required.

Statement Macros

- Example

```
typedef struct
{
  int     emp_num;
  char    name[81];
} EMP_REC_t, *EMP_REC_p_t;
```

- This macro is valid, but not perfect

```
#define INIT_REC( rec_p )            \
  (rec_p)->emp_num = 0;              \
  *(rec_p)->name = ('\0');
  . . .
EMP_REC_t master_rec;
EMP_REC_t other_recs[10];
```

Statement Macros

- This succeeds

```
INIT_REC( &master_rec );
```

- ... but, this doesn't

```
for(     inx = 0     ;
         inx < 10    ;
         inx++
   )
   INIT_REC( &other_recs[inx] );
```

Statement Macros

- **Sometimes statement macros need to be enclosed in a compound statement**

```
#define INIT_REC( rec_p )             \
  {                                   \
   (rec_p)->emp_num = 0;              \
   *(rec_p)->name = NULL_CHAR;        \
  }
```

Related Operators

- Conditional Expression Operator

```
#define LARGER( a, b )    \
            ( (a) > (b) ? (a) : (b) )
int inx = 99,
    jnx = 14;
printf( "Larger = %d\n",
            LARGER( inx, jnx ) );
```

Related Operators

- Comma Operator

```
#define MY_MALLOC( size )     \
 (                            \
   puts( "Calling malloc" ), \
   malloc( (size) )          \
  )
#define MY_FREE( ptr )        \
 (                            \
   puts( "Calling free" ),   \
   free( (ptr) ),            \
   (ptr) = NULL              \
 )
```