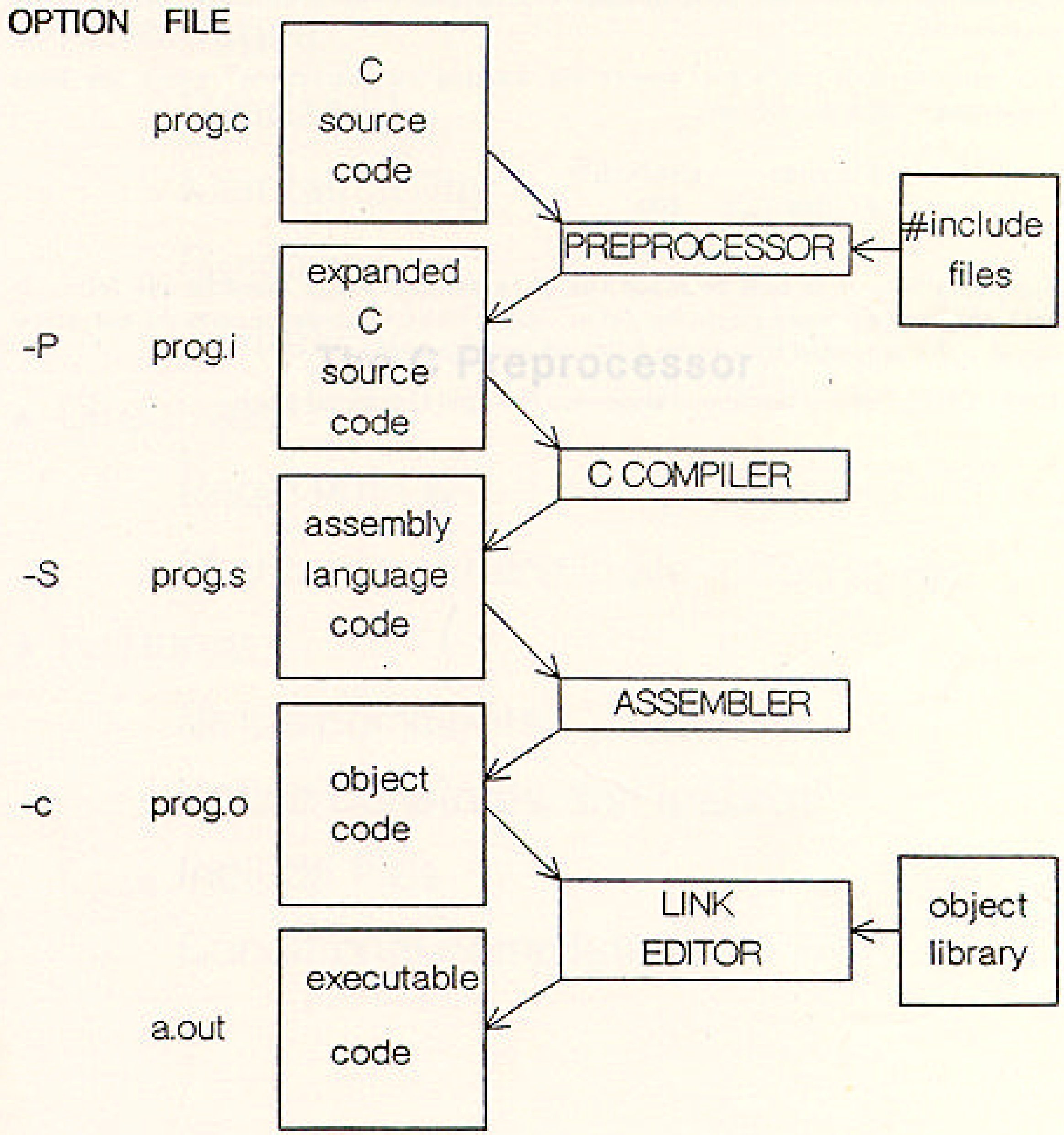
The C Preprocessor and Libraries

Unit 4 Objectives

- Describe the compilation steps
- Use preprocessor directives in a program
- Use existing library functions

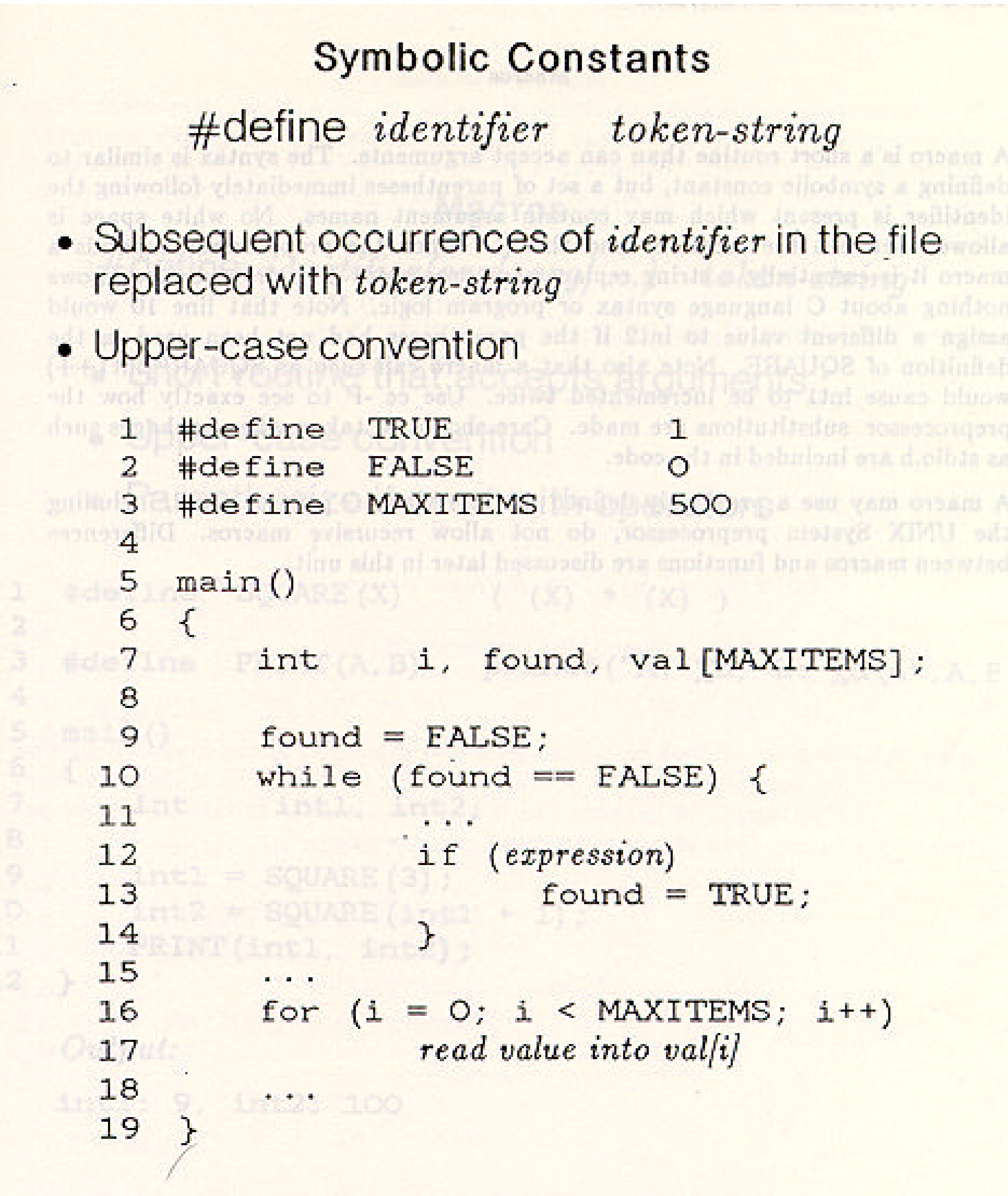
C Compilation Steps and UNIX System cc(1) Options



The C Preprocessor

C Preprocessor Overview

- First compilation step
- Advantages:
  - Readability
  - Maintainability
  - Flexibility
  - Portability
- Directives:
  - Begin with #
  - May be anywhere in file, often at top
- Features:
  - Strips comments
  - Define constants and macros
  - Include files
  - Conditional compilation

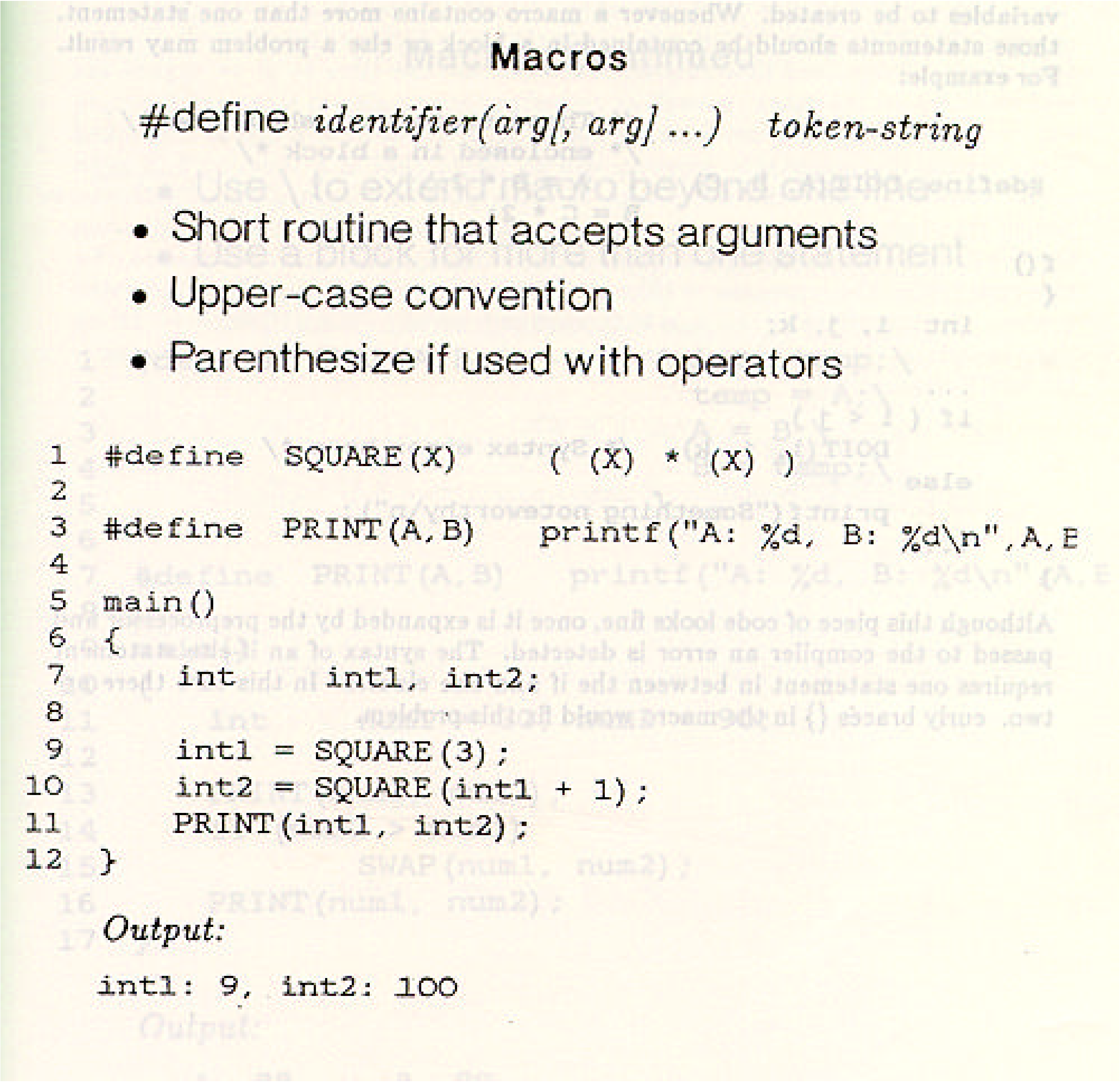


Symbolic Constants

#define *identifier* *token-string*

- Subsequent occurrences of *identifier* in the file replaced with *token-string*
- Upper-case convention

```
#define  TRUE          1
#define  FALSE         0
#define  MAXITEMS      500

main()
{
        int      i,  found,  val[MAXITEMS];

        found = FALSE;
        while (found == FALSE) {
                ...
                if  (expression)
                        found = TRUE;
                }
        ...
        for (i = 0; i < MAXITEMS; i++)
                read value into val[i]
        ...
}
```



Macros

#define *identifier(arg[, arg] ...)* *token-string*

- Short routine that accepts arguments
- Upper-case convention
- Parenthesize if used with operators

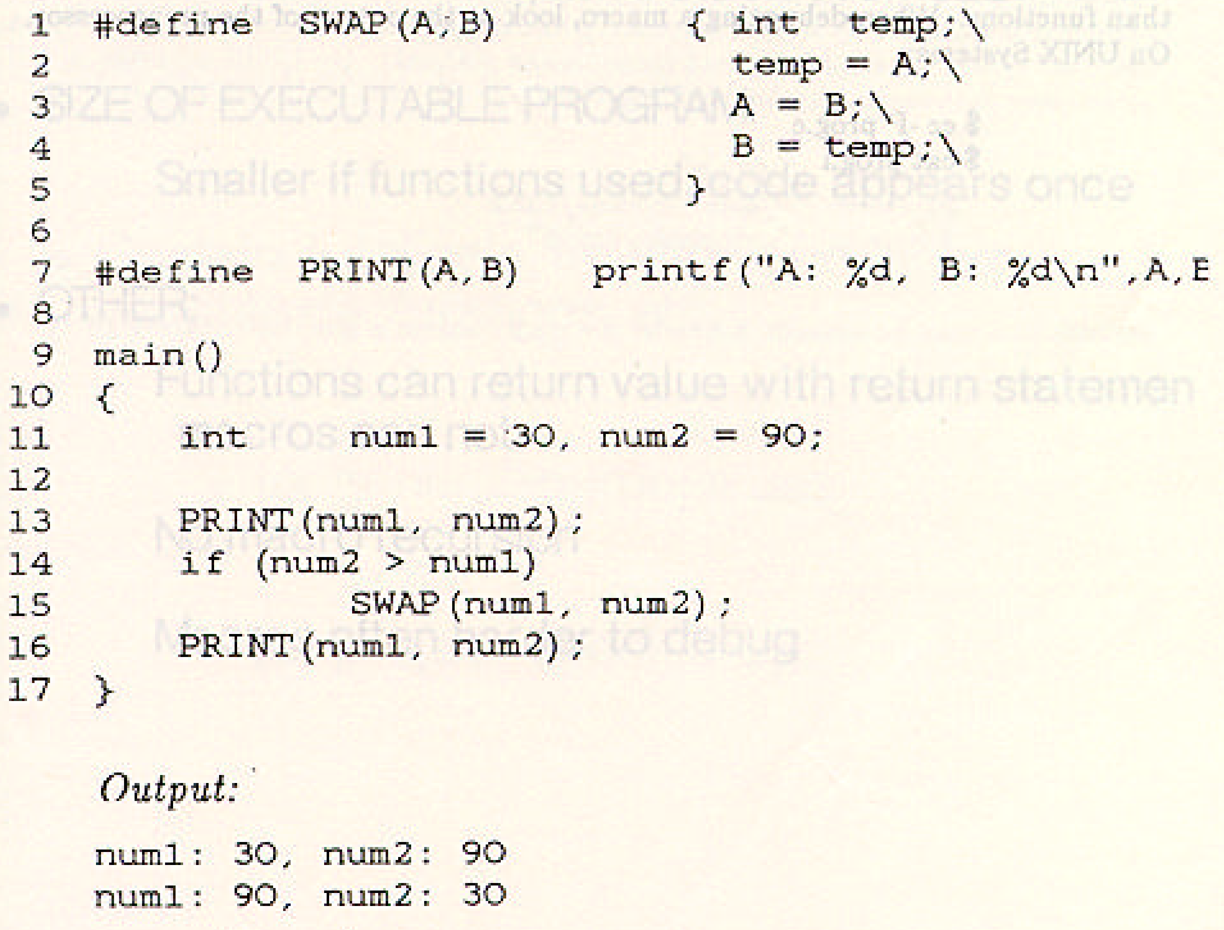
```
#define  SQUARE(X)     ( (X)  * (X) )

#define  PRINT(A,B)   printf("A: %d, B: %d\n",A,B

main()
{
    int     int1, int2;

    int1 = SQUARE(3);
    int2 = SQUARE(int1 + 1);
    PRINT(int1, int2);
}
```

Output:

```
int1: 9, int2: 100
```

Macros, Continued

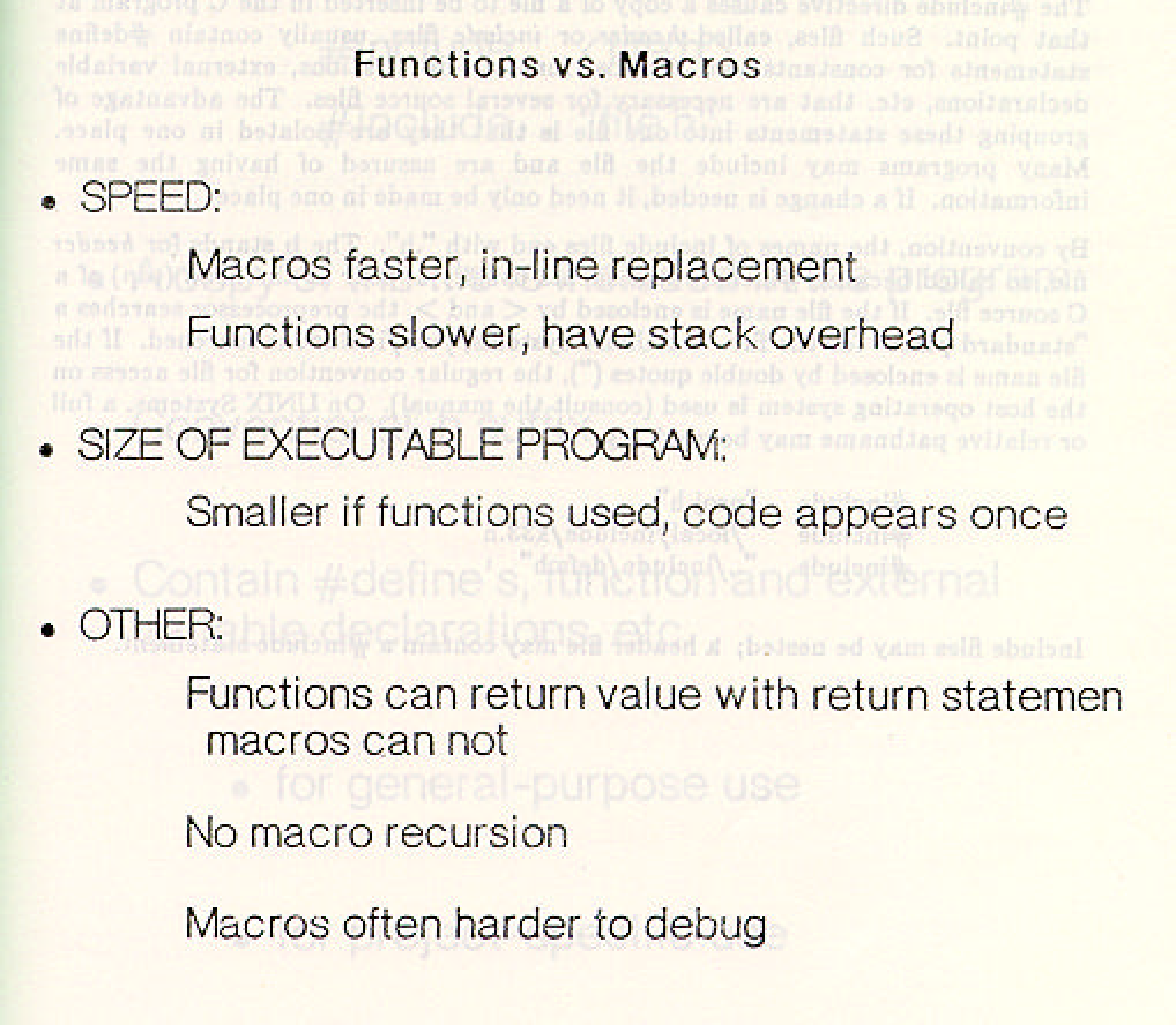
- Use \ to extend macro beyond one line
- Use a block for more than one statement



```
#define  SWAP(A,B)                { int   temp;\
                                    temp = A;\
                                    A = B;\
                                    B = temp;\
                                  }

#define  PRINT(A,B)   printf("A: %d, B: %d\n",A,B

main()
{
    int     num1 = 30, num2 = 90;

    PRINT(num1, num2);
    if (num2 > num1)
            SWAP(num1, num2);
    PRINT(num1, num2);
}
```

Output:

```
num1: 30, num2: 90
num1: 90, num2: 30
```



Functions vs. Macros

- SPEED:

  Macros faster, in-line replacement

  Functions slower, have stack overhead

- SIZE OF EXECUTABLE PROGRAM:

  Smaller if functions used, code appears once

- OTHER:

  Functions can return value with return statemen
  macros can not

  No macro recursion

  Macros often harder to debug

Header Files, Include Files

```
#include    <file.h>
#include    "file.h"
```

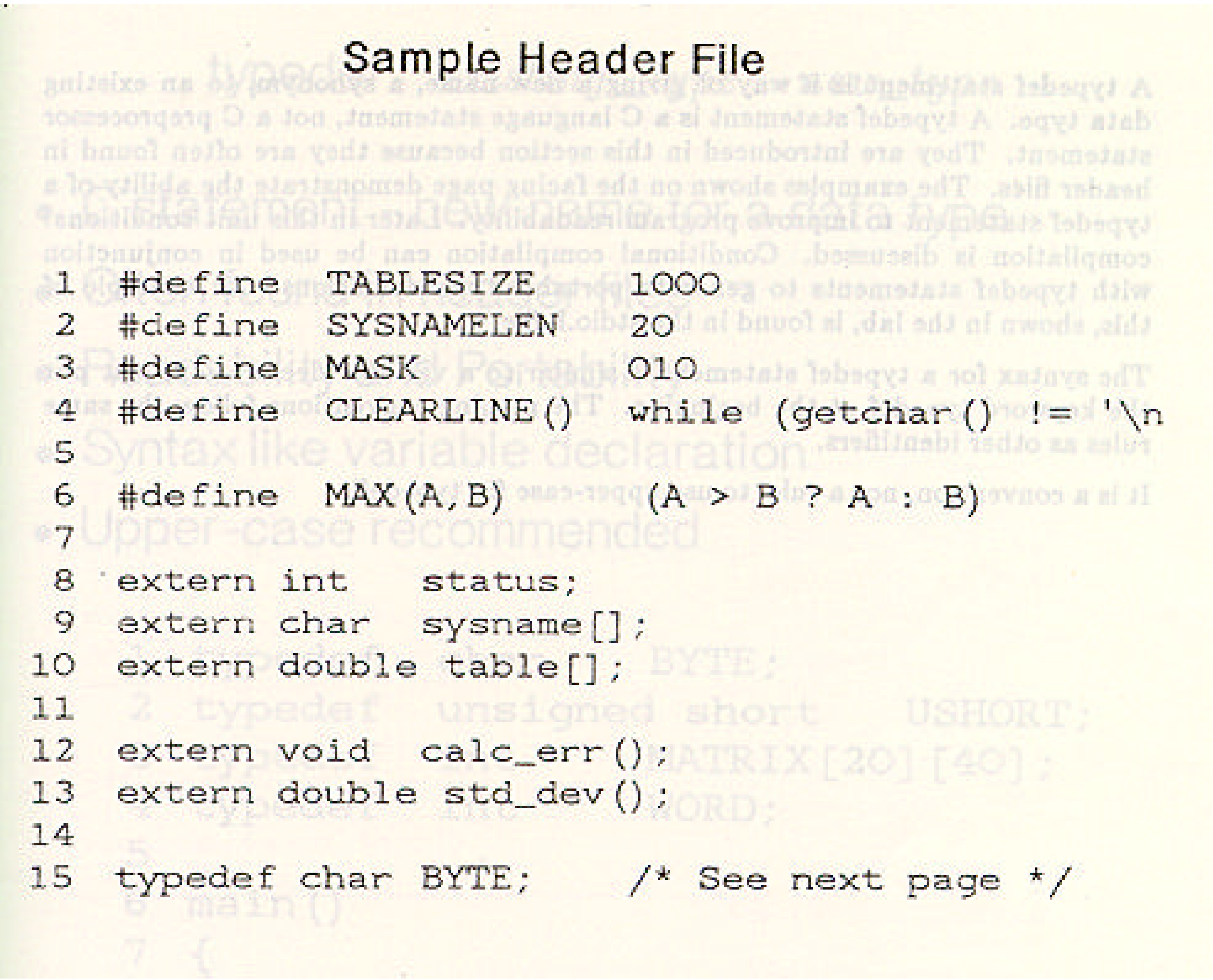
- A copy of the file is included in the program

- Conventional .h suffix

- Contain #define's, function and external variable declarations, etc.

  - for general-purpose use

  - for project-specific use



Sample Header File

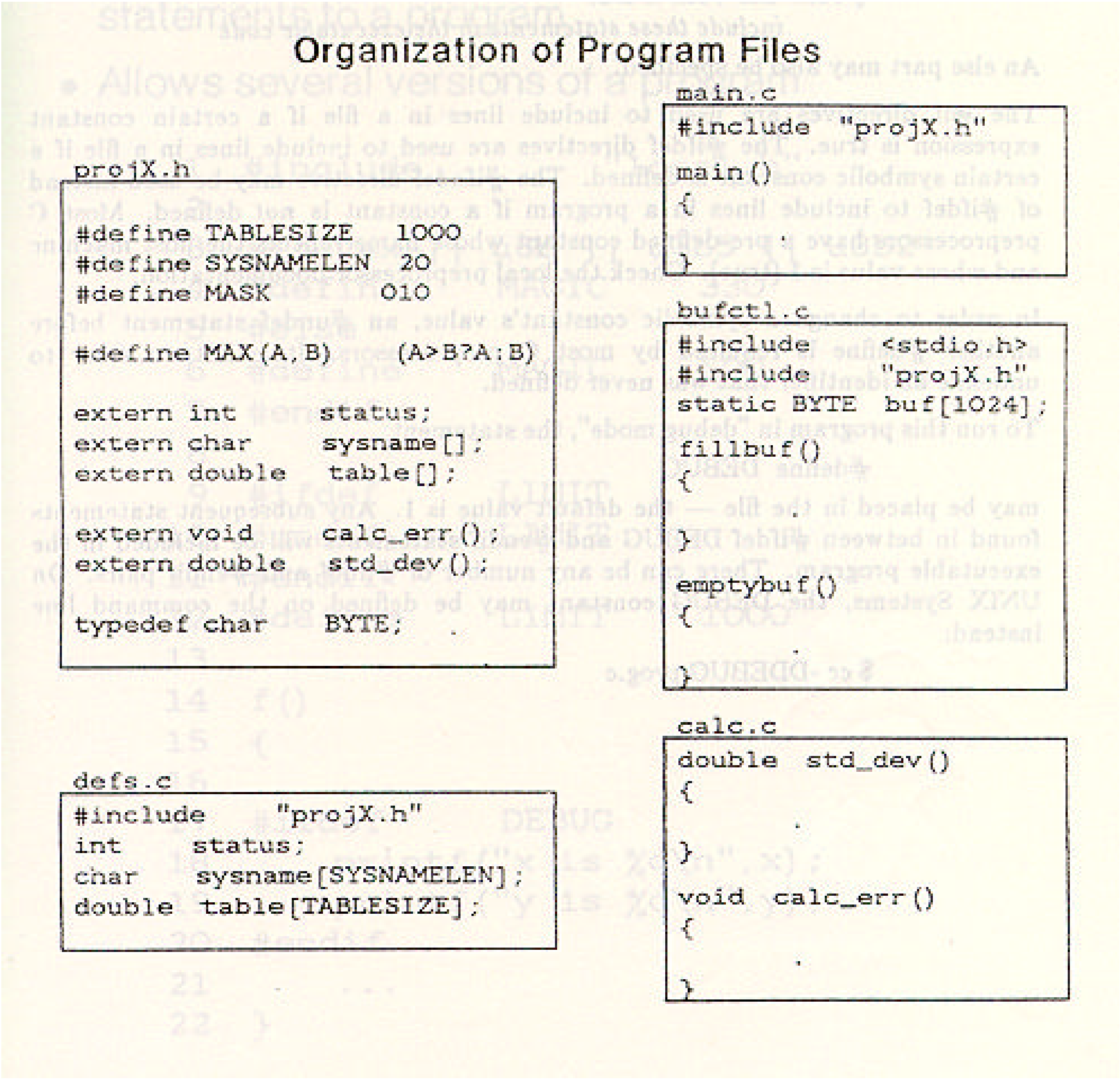
```
#define   TABLESIZE     1000
#define   SYSNAMELEN    20
#define   MASK          010
#define   CLEARLINE()   while (getchar() != '\n

#define   MAX(A,B)            (A > B ? A : B)

extern int    status;
extern char   sysname[];
extern double table[];

extern void   calc_err();
extern double std_dev();

typedef char BYTE;     /* See next page */
```

Renaming a Type - typedef

typedef *existing_type* *new_type*

- C statement - new name for a data type
- Often found in header files
- Readability and Portability
- Syntax like variable declaration
- Upper-case recommended

```
typedef  char     BYTE;
typedef  unsigned short   USHORT;
typedef  int      MATRIX[20][40];
typedef  int      WORD;

main()
{
    BYTE     input;
    WORD     buf[512];
    MATRIX   prev, current;

    ...
```



Organization of Program Files

projX.h

```
#define TABLESIZE   1000
#define SYSNAMELEN  20
#define MASK        010

#define MAX(A,B)    (A>B?A:B)

extern int      status;
extern char     sysname[];
extern double   table[];

extern void     calc_err();
extern double   std_dev();

typedef char    BYTE;
```

defs.c

```
#include    "projX.h"
int     status;
char    sysname[SYSNAMELEN];
double  table[TABLESIZE];
```

main.c

```
#include   "projX.h"
main()
{
        :
}
```

bufctl.c

```
#include      <stdio.h>
#include      "projX.h"
static BYTE  buf[1024];
fillbuf()
{
        :
}
emptybuf()
{
        .
}
```

calc.c

```
double  std_dev()
{
        .
}
void  calc_err()
{
        .
}
```

Conditional Compilation

#if, #ifdef, #ifndef

- Conditionally adds C and/or preprocessor statements to a program
- Allows several versions of a program

```
#include        "local.h"

#if vax || u3b || u3b5 || u3b2
#define     MAGIC     330
#else
#define     MAGIC     500
#endif

#ifdef      LIMIT
#undef      LIMIT
#endif
#define     LIMIT     1000

f()
{
        ...
#ifdef      DEBUG
    printf("x is %d\n",x);
    printf("y is %d\n",y);
#endif
        ...
}
```

Exercise – Preprocessor

1. The C compiler finds an error on line 5. Why?

```
1       #define  LINELEN  80 ;
2
3       main()
4       {
5               char          line[LINELEN] ;
6               int           x ;
7               •
```

2. What is wrong with the ISDIGIT macro? Fix the macro.

```
#define  ISDIGIT(C)    return((C) >= ‘0’ && (c) <= ‘9’) ;

main()
{
            int            input, digits = 0 ;

            input = getchar() ;
            if (ISDIGIT(input))
                    digits ++ ;
            •
```

Libraries

What is a Library?

- Collection of shared functions
- Supplied with system or created by programmer
- Library functions usually supplied with C compiler

  Input/output - “Standard I/O”

  String handling

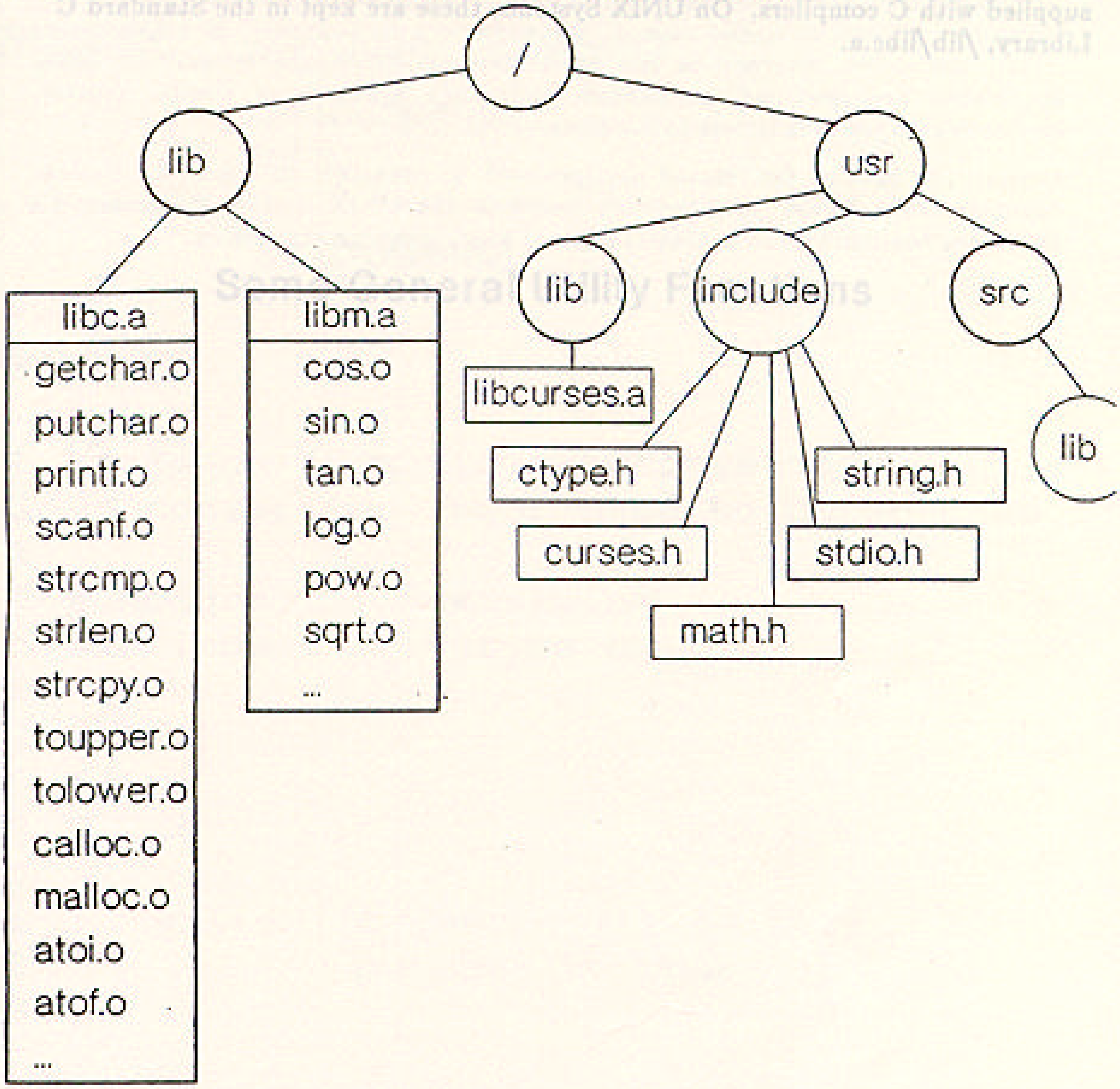
  Character handling

  Memory allocation

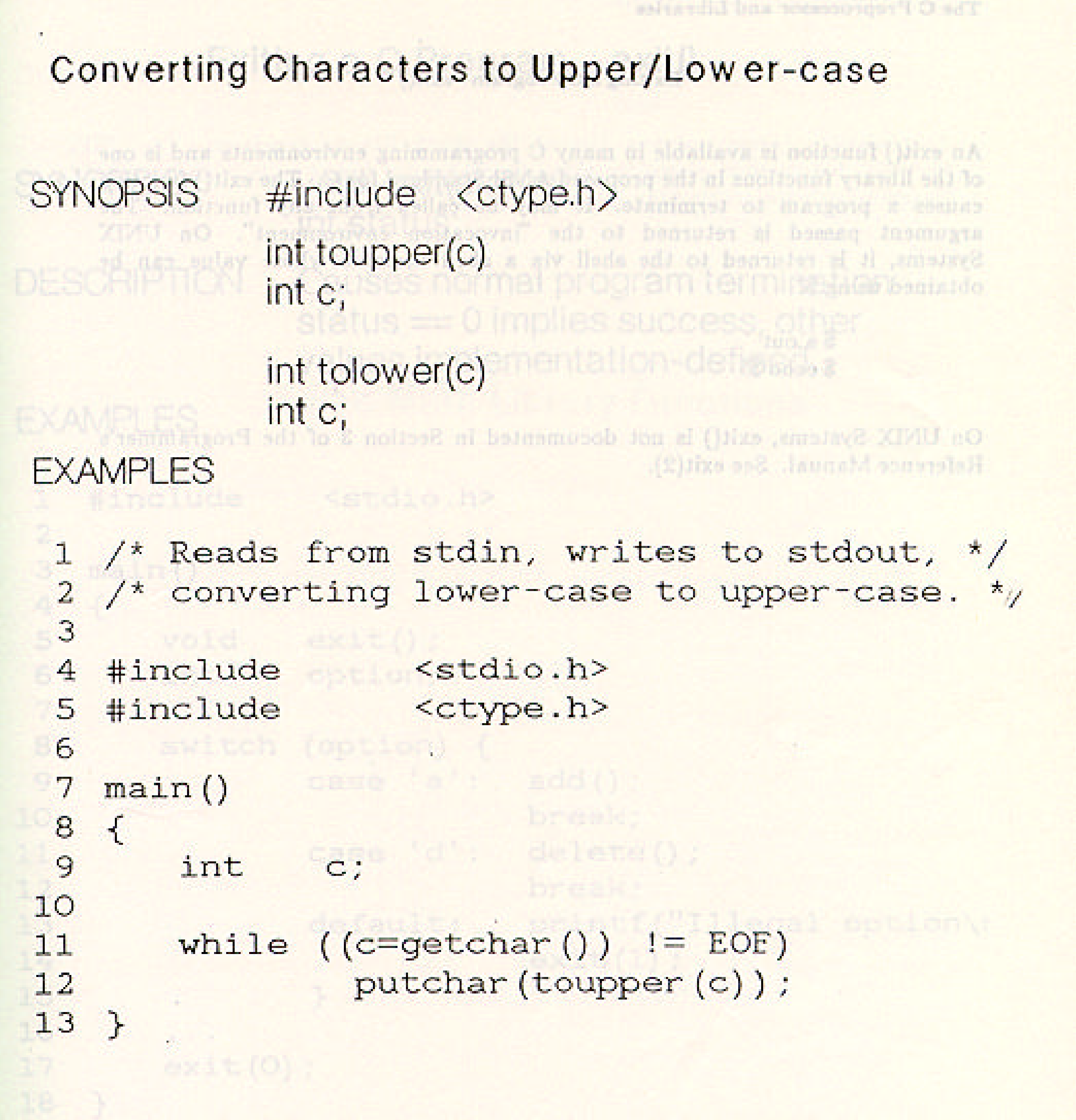
  General utilities

  Math functions

Libraries on the UNIX System



Some General Utility Functions



Converting Characters to Upper/Lower-case

SYNOPSIS

```
#include   <ctype.h>
int toupper(c)
int c;

int tolower(c)
int c;
```

EXAMPLES

```
/* Reads from stdin, writes to stdout, */
/* converting lower-case to upper-case. */

#include        <stdio.h>
#include        <ctype.h>

main()
{
    int    c;

    while ((c=getchar()) != EOF)
             putchar(toupper(c));
}
```

Exiting a C Program - exit()

SYNOPSIS **void exit(status)**
int status ;

DESCRIPTION **Causes normal program termination.**
status == 0 implies success, other
values implementation-defined.

EXAMPLES

```
#include    <stdio.h>

main()
{
         void       exit() ;
         int        option ;
         • • •
         switch (option) {
                    case  ‘a’:  add() ;
                                break ;
                    case  ‘d’:  delete() ;
                                break ;
                    default :   printf(“Illegal option\n”) ;
                                exit(1) ;
                    }
         • • •
         exit(0) ;
 }
```

Some Math Library Functions

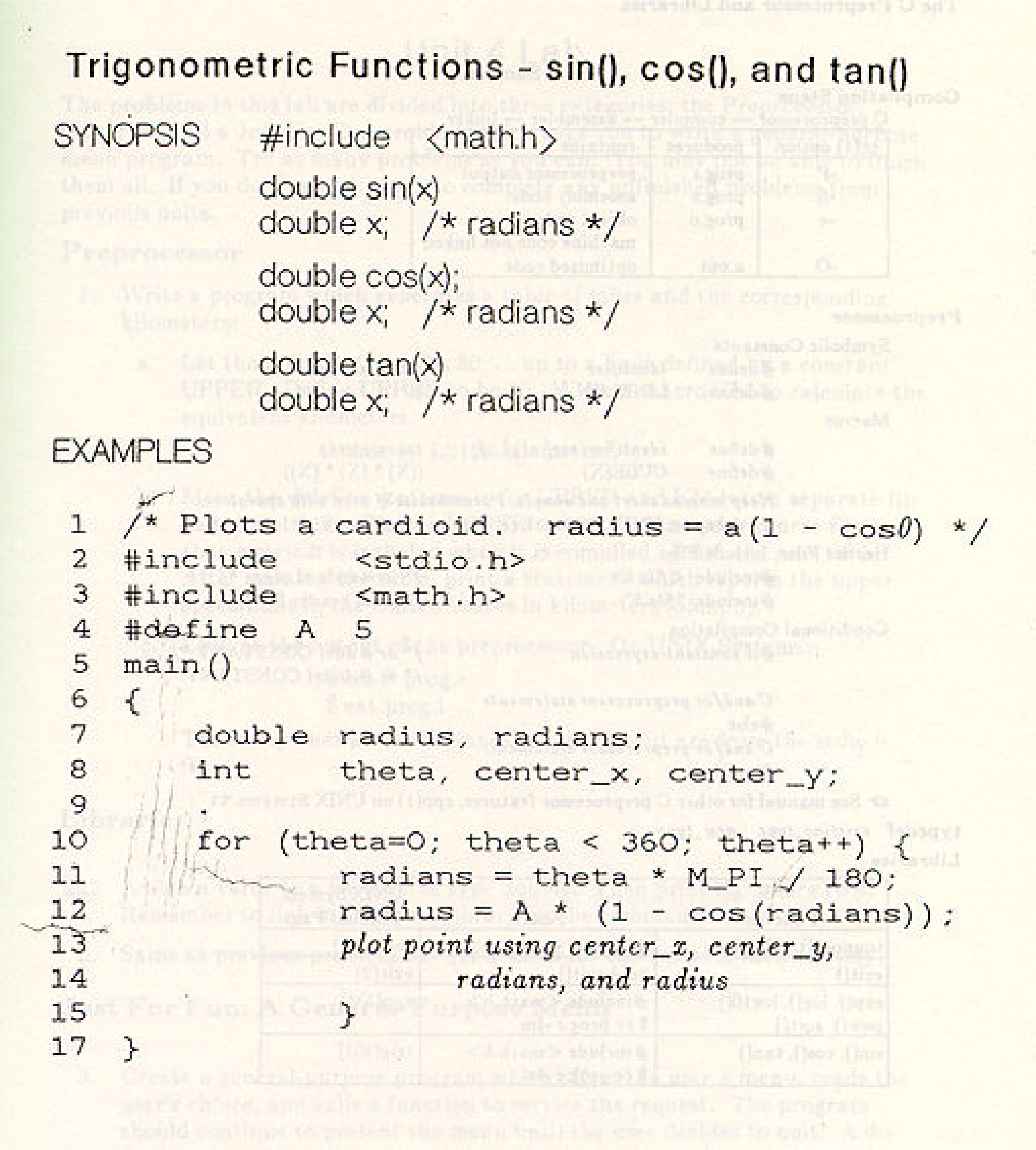
Exponential, Log, Power, and Square Root Functions

SYNOPSIS

```
#include   <math.h>
double exp(x)           /* $e^x$ */
double x ;

double log(x)           /* natural log of x */
double x ;

double log 10(x)        /* log base 10 of x */
double x ;

double pow(x,y)         /* $x^y$ */
double x,y ;

double sqrt(x)          /* square root of x */
double x ;
```

EXAMPLES

```
#include   <stdio.h>
#include   <math.h>
main()
{
        double  z = 77.9 ;
        printf(“ %g    %g    ”, exp(z), log(z)) ;
        printf(“ %g    %g    ”, log10(z), pow(z, 5.0)) ;
        printf(“ %g\n”, sqrt(z)) ;
}
$  cc  prog.c  -lm
$  a.out
6.78485e+33    4.35543    1.89154    2.86871e+09    8.8261
```



Trigonometric Functions - sin(), cos(), and tan()

SYNOPSIS

```
#include  <math.h>

double sin(x)
double x;   /* radians */

double cos(x);
double x;   /* radians */

double tan(x)
double x;   /* radians */
```

EXAMPLES

```
/* Plots a cardioid.  radius = a(1 - cosθ) */
#include     <stdio.h>
#include     <math.h>
#define  A  5
main()
{
    double  radius, radians;
    int     theta, center_x, center_y;

    for (theta=0; theta < 360; theta++) {
            radians = theta * M_PI / 180;
            radius = A * (1 - cos(radians));
            plot point using center_x, center_y,
                  radians, and radius
            }
}
```