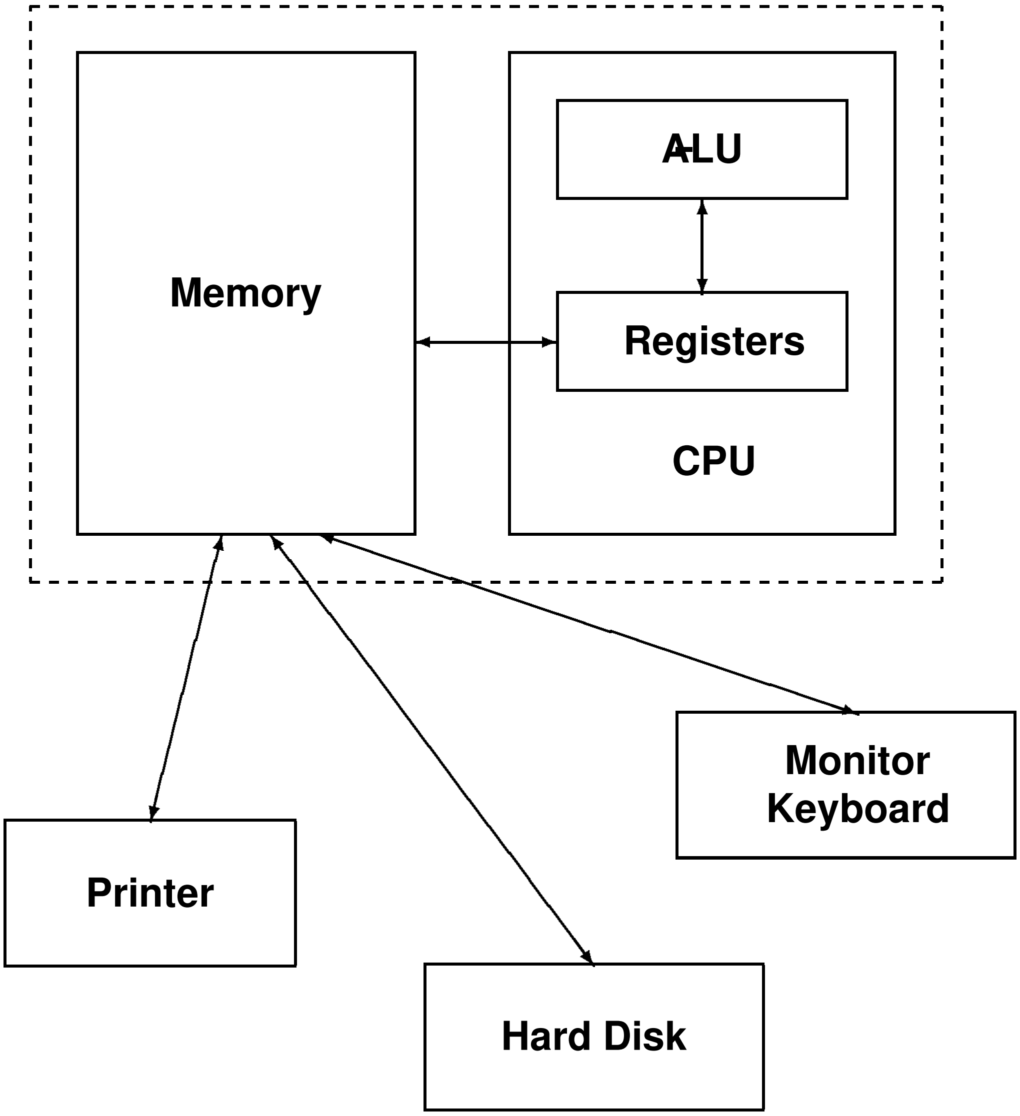
CHAPTER 1

GETTING STARTED WITH C

COMPUTER PROGRAMMING

- A computer is a machine that executes computer programs.
- A computer program is a sequence of instructions.
- Computer programming is the art of constructing computer programs.

A STORED PROGRAM COMPUTER



PROGRAMMING LANGUAGES

- Computers understand machine language programs.
- Machine language programs do very low-level tasks:
  - Move data from memory to a register.
  - Add a value to a register.
  - Compare two values for equality.
- C is a high-level language.
- A single high-level language instruction may turn into a sequence of machine language instructions.
- Higher-level language programs are translated into machine language by a *compiler*.

Sample Output – welcome

```
Welcome to the Joy of C!
```

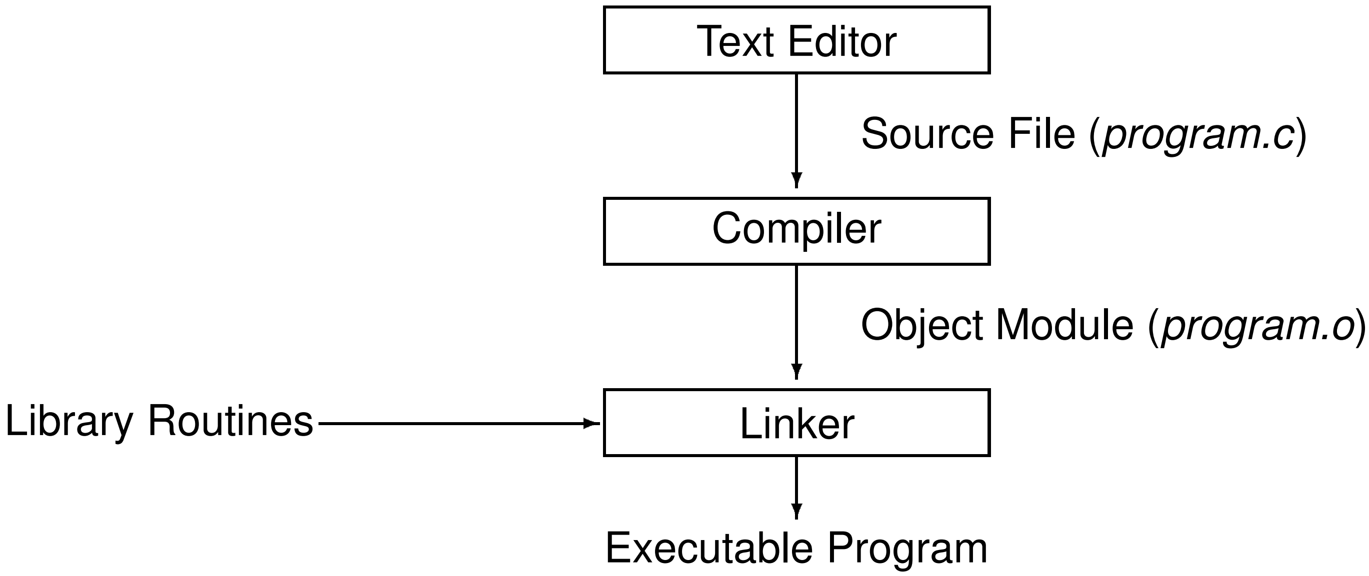
welcome.c

(Figure 1.1 – Page 4)

```
/*
 * A program that prints a welcoming message.
 */

main()
{
  printf("Welcome to the Joy of C!\n");
}
```

MAKING AN EXECUTABLE



COMPILING ON UNIX

- Use the *cc* command to compile and link the program.

  cc welcome.c

- By default, this creates an executable named *a.out*.

- Use the *-o* option to specify the executable's name.

  cc -o welcome welcome.c

- To run the program, simply type the executable's name.

  welcome

- On some systems you may have to add extra options (such as *-Aa*) to request an ANSI C compilation.

  cc -Aa -o welcome welcome.c

welcome2.c

(Figure 1.3 – Page 7)

```
/*
 * An improved version of our program to print a welcoming message.
 */
#include <stdio.h>

main()
{
  printf("Welcome to the Joy of C!\n");
  return 0;
}
```

Sample Output – price

```
List price per item: 13.950000
List price of 23 items: 320.850000
Price after 18.000000% discount: 263.097000
Sales tax at 7.500000%: 19.732275
Total cost: 282.829275
```

price.c
(Figure 1.4 – Page 9)

```
/*
 * Compute the actual cost of buying 23 items at $13.95.  We assume an
 * 18% discount rate and a 7.5% sales tax.
 */
#include <stdio.h>

main()
{
  int    items;                         /* # of items bought */
  double list_price;                    /* list price of item */
  double discount_rate;                 /* discount percentage */
  double sales_tax_rate;                /* sales tax percentage */
  double total_price;                   /* total price of all items */
  double discount_price;                /* discount price of all items */
  double sales_tax;                     /* amount of sales tax */
  double final_cost;                    /* cost including sales tax */

  items = 23;
  list_price = 13.95;
  discount_rate = .18;
  sales_tax_rate = .075;

  total_price = items * list_price;
  discount_price = total_price - total_price * discount_rate;
  sales_tax = discount_price * sales_tax_rate;
  final_cost = discount_price + sales_tax;

  printf("List price per item: %f\n", list_price);
  printf("List price of %i items: %f\n", items, total_price);
  printf("Price after %f%% discount: %f\n",
         discount_rate * 100, discount_price);
  printf("Sales tax at %f%%: %f\n",
         sales_tax_rate * 100, sales_tax);
  printf("Total cost: %f\n", final_cost);

  return 0;
}
```

newprice.c

(Not In Text)

```
/*
 * A version of the price-computing program that minimizes the use
 * of variables.  It's much harder to understand.
 */
#include <stdio.h>

main()
{
  double total_price;              /* total price of all items */
  double discount_price;           /* discount price of all items */
  double sales_tax;                /* amount of sales tax */
  double final_cost;               /* cost including sales tax */

  total_price = 23 * 13.95;
  discount_price = total_price - total_price * .18;
  sales_tax = discount_price * .075;
  final_cost = discount_price + sales_tax;

  printf("List price per item: %f\n", 13.95);
  printf("List price of %i items: %f\n", 23, total_price);
  printf("Price after %f%% discount: %f\n",
         .18 * 100, discount_price);
  printf("Sales tax at %f%%: %f\n",
         .075 * 100, sales_tax);
  printf("Total cost: %f\n", final_cost);

  return 0;
}
```

Sample Output – price2

```
List price per item:               13.95
Total price of 23 items:          320.85
Price after 18.0% discount:       263.10
Sales tax at 7.5%:                 19.73
Total cost:                       282.83
```

price2.c

(Figure 1.5 – Page 13)

```
/*
 * An improved version of our cost-computing program.
 */
#include <stdio.h>

#define ITEMS           23
#define LIST_PRICE      13.95
#define DISCOUNT_RATE     .18
#define SALES_TAX_RATE    .075

main()
{
  double discount_price;           /* discounted price of item */
  double total_price;              /* total price of all items */
  double sales_tax;                /* amount of sales tax */
  double final_cost;               /* cost including sales tax */

  total_price = ITEMS * LIST_PRICE;
  discount_price = total_price - total_price * DISCOUNT_RATE;
  sales_tax = discount_price * SALES_TAX_RATE;
  final_cost = discount_price + sales_tax;

  printf("List price per item:\t\t%7.2f\n", LIST_PRICE);
  printf("List price of %i items:\t\t%7.2f\n", ITEMS, total_price);
  printf("Price after %.1f%% discount:\t%7.2f\n",
         DISCOUNT_RATE * 100, discount_price);
  printf("Sales tax at %.1f%%:\t\t%7.2f\n",
         SALES_TAX_RATE * 100, sales_tax);
  printf("Total cost:\t\t\t%7.2f\n", final_cost);

  return 0;
}
```

messy.c

(Figure 1.6 – Page 16)

```
/*
 * A very messy version of our cost-computing program.
 */
#include <stdio.h>
main() {double dp;double tp;double st;double fc;
tp=23*13.95;dp=tp-tp*.18;st=dp*.075;fc=dp+st;
printf("List price per item:\t\t%7.2f\n",13.95);
printf("List price of %i items:\t\t%7.2f\n",23,tp);
printf("Price after %.1f%% discount:\t%7.2f\n",.18*100,dp);
printf("Sales tax at %.1f%%:\t\t%7.2f\n",.075*100,st);
printf("Total cost:\t\t\t%7.2f\n",fc);return 0;}
```

REMINDERS

- When you use a predefined function, be sure to include the appropriate header file.

- Always return a value from `main` indicating whether the program succeeded or failed.

- Make sure there's a one-to-one correspondence between the formatting instructions and values to print.

CHAPTER SUMMARY

- A C program consists of several parts:

  preprocessor statements

  ```
  main()
  {
     declarations
     statements
  }
  ```

- Anything between a `/*` and `*/` is treated as a comment and ignored by the compiler.

- All C programs are run through a preprocessor before they are passed to the compiler proper. Preprocessor statements are lines starting with a `#`. There are preprocessor statements to include system header files (`#include`) and to give symbolic names to constants (`#define`).

- We declare variables by providing a type (`int` for integers or `double` for floating point numbers) and a name. We assign values to variables using the = operator.

- C provides the usual arithmetic operators: `+` for addition, `–` for subtraction, `*` for multiplication, and `/` for division. The precedence of `*` and `/` is higher than that of `+` and `–`.

CHAPTER SUMMARY

- We perform output by calling the **`printf`** library function.

  `printf("`*format string***`",`** *expression-1*, *expression-2*, . . .)

  It uses special formatting codes in its first parameter to determine how to display subsequent parameters.

- **`main`** uses the **`return`** statement to provide an indication of success or failure. By convention, a 0 indicates success, and anything else indicates failure.