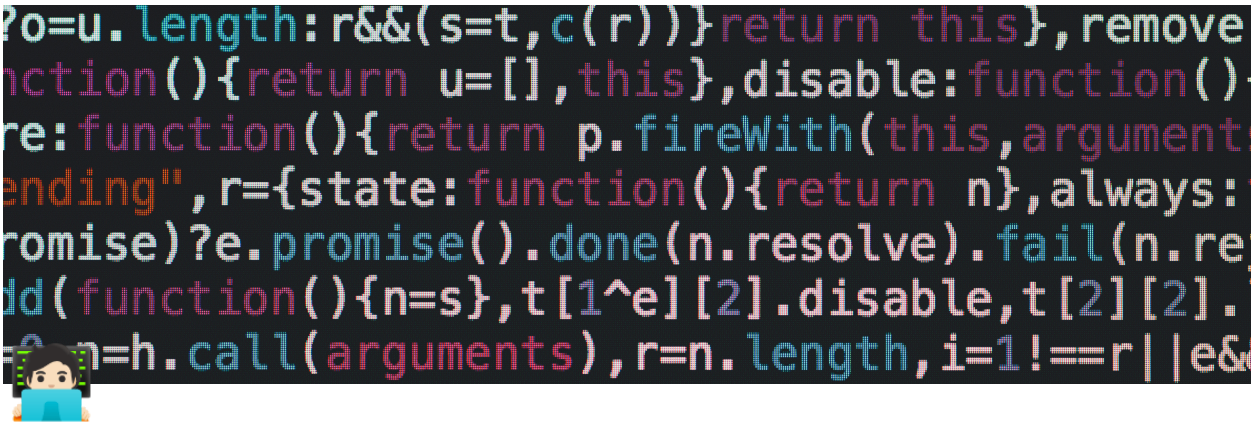


Intro to C #1057

Grading

Aa Title	Column 1
Labs	15
Quizzes	15
Midterm	30
Final	40

▼ **Chapters**

1. Chapter 1
2. Chapter 2
3. Chapter 3
4. Chapter 4
5. Chapter 5

Chapter 1

Intro/Computers and Programming

▼ Lecture PDF

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/946045dd-1860-4bb2-af2c-97300d8e09cf/CIS1057Ch01.pdf

▼ Basic Components of Computers

- CPU
- RAM
- Secondary Storage
- *Motherboard
- Input devices
- Output devices

▼ Computer Languages

- Instructions for a computer to execute
- Machine language (binary)
- Assembly language (instructions that are exe on processor)
- High-level language

– C, C++, Objective C, Java, VB, C#, etc.

– Word processor or IDE

- Making an executable (.exe)

– Source file

– Compiler

– Object file

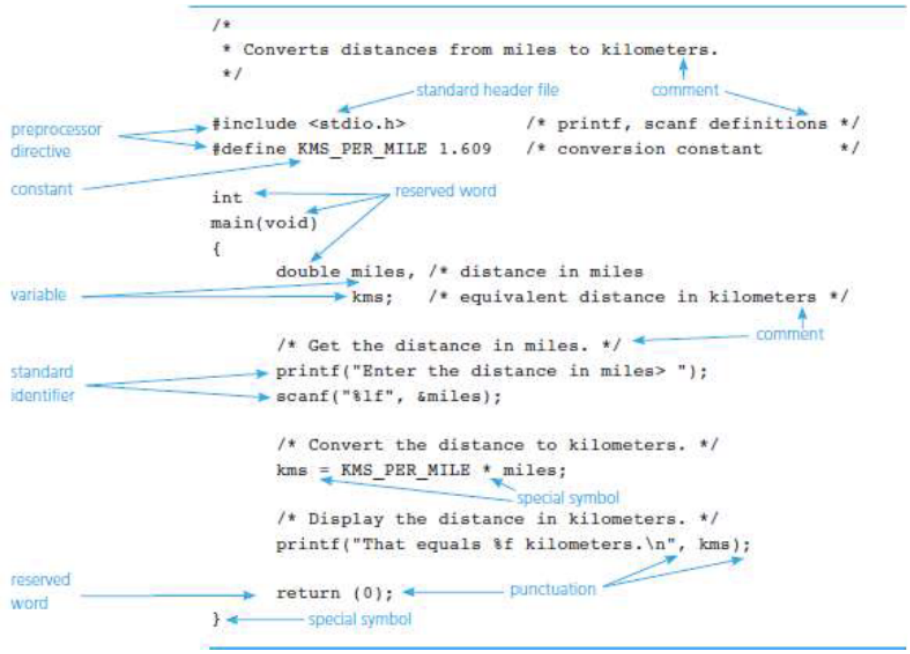
Chapter 2

Overview of C

▼ Lecture PDF

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4355321d-c064-42df-9016-933c82b2f774/CIS1057Ch02.pdf

Figure 2.1 **C Language Elements in Miles-to-Kilometers Conversion Program**



1-3

▼ Preprocessor Directives

- #include statements include internal C and programmer defined libraries with useful data and functions
  - #include <stdio.h>
  - #include <math.h>
  - #include "myHeader.h"
- #define statements facilitate the definition of constants and macros
  - #define PI 3.14159
  - #define ADD(a, b) (a +

```
main() function
int main(int argc, char *argv[]){
// Variable declarations
// Executable statements
```

```
return 0;
}
```

▼ Variable Declarations

- ▼ Int

  Whole numbers

  for example i = 1

- ▼ Double

  Decimal numbers

  for example i = 1.50

- ▼ Char
  - 8-bit character
  - Numerically -128 to 127 or 0 to 255(unsigned)
  - ASCII character set
    - ‘ ‘ = 32
    - ‘0’ = 48
    - ‘9’ = 57
    - ‘A’ = 65
    - ‘Z’ = 90
    - ‘a’ = 97
    - ‘z’ = 122
- ▼ Short
  - 16-bit integer
    - -32,768 to 32,767
    - unsigned short
  - 0 – 65,535
    - int (or long) – 32-bit integer
  - -2,147,483,648 to 2,147,483,647
    - unsigned int (or long) – unsigned integer
  - 0 to 4,294,967,295
- ▼ Float
  - 32-bit floating point
  - ±10-37 to 1038
  - 6 significant digits
    - double – 64-bit floating point
  - ±10-307 to 10308

- 15 significant digits
  - long double – 80-bit floating point
- ±10-4931 to 104932
- 19 significant digits

▼ Casting Variables(converting)

- Convert value in one variable to another

```
- double dbl = 150.123;
- int i = (int) dbl;
- i = ?(150)
```

Figure 2.2 **Internal Format of Type int and Type double**

▼ Executable Statements

```
printf("Hello World\n");
scanf("%d", &variable);
  c = a + b;
  c = pow(a,b);
```

printf()

```
// Print formatted output
printf("Hello World!\n");
printf("The int is: %d\n", n);
printf("The float is: %f\n", f);
printf("The char is: %c\n", c);
// Placeholders
 %d   %f   %s   %c
// Escape sequence for special chars
 \n  \t  \"  \'
```

scanf()

```
// Reads data from command line
scanf(“%s”, name);
scanf(“%d”, &age);
scanf(“%s %d”, name, &age);
// Placeholders
  %d   %f   %lf   %s   %c
```

- Class Example Code
  - Ch02LecCode.c

```
#include <stdio.h>

#define PI 3.14159

int main(int argc, char *argv[]){
  int anInt; // Declare integer
  double aDouble = 25.5; // Declare double
  float aFloat = 0; // Declare float
  char aChar = 'a'; // Declare char
  char aString[] = "Hello World!"; // Declare string
  double tax_rate_1996; // Declare double
  unsigned int uInt = 1000; // Declare unsigned integer

  double sum = 0; // Declare double
  int n = 5; // Declare integer
  double x = 30.45; // Declare double

  // Perform division with double and float
  sum = aDouble/aFloat;
  printf("sum = %f\n", sum);
  // Perform division with int casted as double, a double and a double literal
  sum = (double)n + x * 5.0;
  printf("The sum is %f\n", sum);

  // These printf calls show the syntax for this function
  // Note that the placeholders are different for most data types
  printf("anInt =  %d %f %f\n", anInt, aFloat, aDouble);
  printf("aFloat =  %1.3f\n", aFloat);
  printf("aDouble =  %1.3f\n", aDouble);
  printf("aChar =  %c\n", aChar);
  printf("aString =  %s\n", aString);
  printf("o =  %c\n", aString[4]);

  // Print string literal "Hello"
  printf("Hello\n");
  // Print command line arguments
  printf("argv[0] = %s\n", argv[0]);
  if(argc > 1)
    printf("argv[1] = %s\n", argv[1]);
  printf("There are %d arguments\n", argc);

  // Get a double from a user
  printf("Enter a double: ");
  scanf("%lf", &aDouble);
  printf("You entered %lf\n", aDouble);

  // Get two strings (words) from a user
  char aString2[100];
  printf("Enter two strings: ");
  scanf("%s %s", aString, aString2); // Get two words seperated by a space
  printf("You entered %s %s\n", aString, aString2);

  // Get a line of text from a user
  printf("Enter a line: ");
```

```
  fflush(stdin); // This clears the input data stream
  gets(aString2); // Get line of text
  printf("%s\n", aString2);

  // Perform division and modulus
  int w  = 37;
  int z = w / n;  // Perform division
  int y = w % n;  // Find remainder
  // Print results
  printf("z = %d y = %d\n", z, y);

  // Calculate coins to be given for change
  int amount = 0, init_amount = 0;
  int quarters, dimes, nickels, pennies;
  // Get amount from user
  printf("Enter a value between 1 and 99: ");
  scanf("%d", &amount);
  init_amount = amount;

  // Calculate coins
  quarters = amount / 25;
  amount = amount % 25;
  dimes = amount / 10;
  amount = amount % 10;
  nickels = amount / 5;
  amount = amount % 5;
  pennies = amount;

  // Print results
  printf("The change for %d cents is: \n %d quarters\n %d dimes\n %d nickels \n %d pennies\n",
  init_amount, quarters, dimes, nickels, pennies);

  // Read three ints from user
  printf("Enter 3 integers seperated by spaces: ");
  scanf("%d %d %d", &w, &y, &z);
  printf("%d %d %d\n", w, y, z);  // Print ints
  printf("%d\n", (w+y+z));    // Print sum

  // Change ints values
  w = 5;
  y = 4;
  z = 3;
  // Print ints
  printf("%d %d %d\n", w, y, z);

  return 0;
}


void func(){

}
```

▼ Ch02SampleCode.c

```
#include <stdio.h>    // Include library for standard input and output
#include <math.h>   // Include library for math
#include <limits.h>   // Include library with data type limits
#include <float.h>    // Include library with float limits

#define PI 3.14159    // Define PI as a constant
#define ADD(a,b)(a+b) // Define a macro to add two numbers

void intToBin(int, char[]);   // Prototype for int to binary conversion

// This is a single line comment

/*
  This
  is
  a
```

```
  multi-line
  comment
*/

// The main function is the entry point for the program
// There are three ways to declare a main function
//int main(){
//int main(void){
int main(int argc, char *argv[]){

  // Declare some variables
  int i=0, j=0, k=0;      // Some integers: i, j, and k are usually used as iterators
  double d = 0.0;       // Double for calculations
  char aChar = '\0';      // Declare an 8-bit ASCII character value
  short aShort = 0;     // Declare a 16-bit short
  unsigned short uShort = 0;  // Declare a 16-bit unsigned short
  int anInt = 0;        // Declare a 32-bit integer
  unsigned int uInt = 0;    // Declare an unsigned 32-bit integer
  float aFloat = 0.0;     // Declare a 32-bit floating point number
  double aDouble = 0.0;   // Declare a 64-bit floating point number
  char bin[33];       // To contain a 32-byte binary string
  char aString[50];     // Declare a string to contain 49 chars and a NULL
  char buffer[100];     // String buffer for sprintf
  char first, middle, last; // To store initials
  //const double PI = 3.14159;// Another way to declare a constant

  // Print "Hello World!" to screen
  printf("Hello World!\n");

  // Print command line args to screen
  for(i=0; i<argc; i++)
    printf("argv[%d] = %s\n", i, argv[i]);

  // Convert a string to a integer
  if(argc >= 3){
    i = atoi(argv[2]);
    printf("The integer value of argv[2] is %d\n\n", i);
  }

  // Assign values to data and print to screen with formatting
  // A character
  aChar = 'A';
  printf("The value of the char is: \'%c\'\n\n", aChar);

  // Some shorts - note that signed numbers are %d and unsigned are %u
  aShort = SHRT_MIN;
  printf("The min value of a short is: %16d\n", aShort);
  aShort = SHRT_MAX;
  printf("The max value of a short is: %16d\n", aShort);
  uShort = USHRT_MAX;
  printf("The max value of an unsigned short is: %6u\n\n", uShort);

  // Some integers - note that signed numbers are %d and unsigned are %u
  anInt = INT_MIN;
  printf("The min value of an int is: %22d\n", anInt);
  anInt = INT_MAX;
  printf("The max value of an int is: %22d\n", anInt);
  uInt = UINT_MAX;
  printf("The max value of an unsigned int is: %13u\n\n", uInt);

  // Some floats - note that %e is scientific and %f is floating point
  aFloat = FLT_MIN;
  printf("The min value of a float is: %24e\n", aFloat);
  printf("The min value of a float is: %25.44f\n", aFloat);
  aFloat = FLT_MAX;
  printf("The max value of a float is: %24e\n", aFloat);
  printf("The max value of a float is: %25f\n\n", aFloat);

  // Some doubles - note that %e is scientific and %f is floating point
  aDouble = DBL_MIN;
  printf("The min value of a double is: %23e\n", aDouble);
  printf("The min value of a double is: %1.330f\n", aDouble);
  aDouble = DBL_MAX;
```

```
printf("The max value of a double is: %23e\n", aDouble);
printf("The max value of a double is: %25f\n\n", aDouble);

// Some math expressions with limit values
aShort = 4567;
anInt = 1234567;
// This stores addition of a short and an int in a float
aFloat = aShort + anInt;
printf("%d + %d = %f\n", aShort, anInt, aFloat);
aFloat = 987654.321;
// This stores addition of a short an int and a float in a double
aDouble = aShort + anInt + aFloat;
printf("%d + %d + %f = %f\n", aShort, anInt, aFloat, aDouble);
// This stores addition of a float and an int in a double
anInt = aFloat + aDouble;
printf("%f + %f = %d\n", aFloat, aDouble, anInt);
// This overflows the int
anInt = aFloat * aDouble;
printf("%f * %f = %d\n", aFloat, aDouble, anInt);
// Add 1 to the max for an int - overflow warning
anInt = INT_MAX + 1;
printf("(INT_MAX) %d + 1 = %d\n", INT_MAX, anInt);
aFloat = FLT_MAX + 1.0;
printf("(FLT_MAX) %f + 1.0 = %f\n", FLT_MAX, aFloat);
if(aFloat == FLT_MAX)
  printf("The values FLT_MAX and FLT_MAX+1 are the same\n\n");

// Calculate area and circumference of a circle
aFloat = 2.5;     // Store radius in a float
// Calculate area of a circle
aDouble = PI * aFloat * aFloat;
printf("The area of a circle with a radius %f is %f\n", aFloat, aDouble);
// Calculate circumference
aDouble = 2.0 * PI * aFloat;
printf("The circumference of a circle with a radius %f is %f\n\n", aFloat, aDouble);

// Calculate

// Convert int to binary
intToBin(16, bin);
printf("16 in binary is %s\n", bin);
intToBin(INT_MAX, bin);
printf("INT_MAX in binary is %s\n", bin);
intToBin((INT_MAX+1), bin);
printf("INT_MAX+1 in binary is %s\n\n", bin);

// Casting datatypes
// Find the integer portion of a double
anInt = (int)aDouble;
printf("The integer portion of %f is %d\n", aDouble, anInt);
// Find the fractional part of a double
d = aDouble - ((int)aDouble);
printf("The fractional portion of %f is %f\n\n", aDouble, d);

  // Operator precedence
  // First unary: +, -, !, ++ and --
  // Second:      *, / and %
  // Third:       + and  -
  // Last:        =
  anInt = 10;
  aShort = 20;
  aFloat = 5;
  aDouble = anInt + aShort * aFloat;
  printf("%d + %d * %f = %f\n", anInt, aShort, aFloat, aDouble);
  // Parenthesis override precedence
  aDouble = (anInt + aShort) * aFloat;
  printf("(%d + %d) * %f = %f\n", anInt, aShort, aFloat, aDouble);
  // parenthesis are useful for division
  aDouble = (anInt + aShort) / aFloat;
  printf("(%d + %d) / %f = %f\n\n", anInt, aShort, aFloat, aDouble);

  // Specialized assignment statements
  // b = b + a; can be abbreviated to b += a;
```

```
  aDouble = 25;
  aDouble = aDouble + aFloat;
  printf("%f + %f = %f\n", aDouble, aFloat, aDouble);
  aDouble += aFloat;
  printf("%f += %f -> %f\n\n", aDouble, aFloat, aDouble);

// Increment and decrement operators
i = 0;
j = 5;
// Increment j before addition
k = i + (++j);
printf("For k = i + (++j): i=%d j=%d k=%d\n", i, j, k); // k is 6 and j is 6

i = 0;
j = 5;
// Increment j after addition
k = i + (j++);
printf("For k = i + (j++): i=%d j=%d k=%d\n\n", i, j, k); // k is 5 and j is 6

// Show input buffer example from book
// Get an integer
printf("Enter an integer: ");
scanf("%d", &anInt);
printf("You entered %d\n", anInt);

// Get 3 chars
printf("Enter your initials: ");
scanf("%c%c%c", &first, &middle, &last);
printf("You entered %c%c%c\n", first, middle, last);

// Get 3 chars
printf("Enter your initials: ");
scanf("%c", &last);
printf("The last scanf missed the %c\n", last);

// To fix, flush the buffer;
// Get an integer
printf("Enter an integer: ");
scanf("%d", &anInt);
printf("You entered %d\n", anInt);

fflush(stdin);

// Get 3 chars
printf("Enter your initials: ");
scanf("%c%c%c", &first, &middle, &last);
printf("You entered %c%c%c\n", first, middle, last);


// Get data from user
// Get user's name
printf("Enter your name: ");
scanf("%s", aString);
printf("You entered %s\n", aString);

// Get user's age
printf("Enter your age: ");
scanf("%d", &anInt);
printf("You entered %d\n", anInt);

// Get user's money
printf("Enter the amount of money in your pocket: ");
scanf("%lf", &aDouble);
printf("You entered %f\n", aDouble);

// Use sprintf to concatenate the data - sprintf returns the length of the concatenated string
j = sprintf(buffer, "You entered %s %d %f\n", aString, anInt, aDouble);
printf("buffer = %s has %d characters\n", buffer, j); // Has extra carrage return from concatenation
buffer[j-1] = '\0';           // Strip off extra carrage return
printf("buffer = %s has %d characters\n", buffer, j); // Prints on one line

// Get all three from user
printf("Enter your name, age and the amount of money in your pocket: ");
```

```
  scanf("%s %d %lf", aString, &anInt, &aDouble);
  printf("You entered %s %d %f\n", aString, anInt, aDouble);

  // The return statement returns control to OS
  return 0;
}


// Function to convert an int to binary
void intToBin(int n, char bin[33]){
  int i=0, j=0, k=0;
  bin[32] = '\0';
  // Iterate through bits in n
  j = 31;
  for (i = 0; i < 32; i++){
    k = n >> i; // Shift i bits
    if (k & 1)
      bin[j] = '1';
    else
      bin[j] = '0';
    j--;
  }
  return;
}
```

Chapter 3

Functions

▼ Lecture PDF

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/309d1c08-aac2-4a08-8951-313c30d8fbf7/CIS1057Ch03.pdf

▼ Program Development

- Understand Problem
- Analysis
- Data requirements
  - Constants, Input, output and Formulas
- Design
  - Initial algorithm
  - Refine algorithm
- Implementation (Coding)
- Testing
- Maintenance

▼ C Function Libraries

- #include <header.h>
  - stdio.h
  - math.h
  - string.h
  - stdlib.h
  - ctype.h

▼ Design Considerations

- Top-Down

–Consider the execution of main()

–Map function calls

–Hierarchal diagram

–Stub functions for testing

- Bottom-Up

–Consider functions first

–Analysis and synthesis

–Code and test functions as units

▼ Modular Solution

Modular Solution

- Use a function

```
// Convert kilometers to miles
// Include standard input/output
#include <stdio.h>
// Define conversion constant
#define KMS_PER_MILE 1.609

// Function prototype
void miles_to_kms();

// Main function
int main(){

   miles_to_kms();
   miles_to_kms();
   miles_to_kms();

   return 0;
}
```

```
// Function to get, calculate and print miles to
void miles_to_kms(){
   // Data needed
   // Floating-point variable for miles
   double miles = 0.0;
   // Floating-point variable for kilometers
   double kms = 0.0;

   // Get the distance in miles
   printf("Please enter distance in miles: ");
   scanf("%lf", &miles);

   // Convert the distance to kilometers
   // Kilometers = 1.609 * Miles
   kms = KMS_PER_MILE * miles;

   // Display the distance in kilometers
   printf("%.3f miles is %.3f kilometers.\n", miles, kms);
}
```

Function Example

```
#include <stdio.h>

// Here is a function declaraction
// It is declared as "int", meaning it returns an integer
/*
  Here are the return types you can use:
      char, double, float, int,
        void(meaning there is no return type)
*/
int MyAge() {
  return 25;
}

// MAIN FUNCTION
int main(void) {
  // CALLING THE FUNCTION WE MADE
    MyAge();
}
```

- ▼ Class Example Code
  - ▼ Ch03LecCodeFunctions.c

```
#include <stdio.h>
#define PI 3.14159

// Function prototypes
void printHello();
void printSomething(char[]);
double calcAreaCircle(double);
double calcAreaBox(double, double, double);
char getCharInput();
int getIntInput();
float getFltInput();
double getDblInput();
void getIntNFloatInput(int*, float*);
int getIntBetween1And10(int*);


int main(){

  // Declare data variables
  int anInt = 0;
  float aFloat = 0.0;
  double aDouble = 0.0;
  int anotherInt = 0;

  // Call printHello 4 times
  printHello();
  printHello();
  printHello();
  printHello();

  // Pass "Something" to printSomething
  printSomething("Something");

  // Declare a double
  double radius = 5.5;
  // Print radius, pass radius and the result of the function call to calcAreaCircle
  printf("Radius = %f and area = %f\n", radius, calcAreaCircle(radius));
  // Print literal reals, pass them and print result of call to calcAreaBox
  printf("The sides are: %f %f %f and area = %f\n", 1.0, 2.0, 3.0, calcAreaBox(1.0,2.0,3.0));

  // Call functions to get data from user
  printf("The character is: %c\n", getCharInput());
```

```
  printf("The integer is: %d\n", getIntInput());
  printf("The float is: %f\n", getFltInput());
  printf("The double is: %f\n", getDblInput());

  // Get more than one datum from a function
  // This is passing variables by reference and is not covered until Ch 06
  getIntNFloatInput( &anInt, &aFloat);
  printf("The int is %d and the float is %f\n", anInt, aFloat);
  anotherInt = getIntBetween1And10(&anInt);
  printf("The int is %d and the is %d between 1 and 10\n", anInt, anotherInt);

  return 0;
}

// Function to print hello
void printHello(){
  // Print hello
  printf("Hello\n");
  return;
}

// Function to print a string parameter
void printSomething(char s[]){
  // Print s
  printf("%s\n", s);
  return;
}

// Function to calculate the area of a circle
double calcAreaCircle(double radius){
  // Return area of a circle
  return PI * radius * radius;
}

// Calculate the area of a 3D box
double calcAreaBox(double a, double b, double c){
  // Return area of a box
  return a*b*c;
}

// Get char input from user
char getCharInput(){
  // Char to get input
  char aChar = '\0';  // Initialize char to NULL
  // Get char from user
  printf("Enter a character: ");
  scanf("%c", &aChar);
  // Return char to calling function
  return aChar;
}

// Get integer input from user
int getIntInput(){
  // Integer to get input
  int anInt = 0;    // Initiaize int to 0
  // Get integer from user
  printf("Enter an integer: ");
  scanf("%d", &anInt);
  // Return integer to calling function
  return anInt;
}

// Get float input from user
float getFltInput(){
  // Float to get input
  float aFloat = 0.0;   // Initialize float to 0.0
  // Get float from user
  printf("Enter an float: ");
  scanf("%f", &aFloat);
  // Return float to user
  return aFloat;
}
```

```
// Get double input from user
double getDblInput(){
  // Double to get input
  double aDouble = 0;   // Initialize double to 0.0
  // Get double from user
  printf("Enter an double: ");
  scanf("%lf", &aDouble);
  // Return double to user
  return aDouble;
}

// Get integer input from user
// Notice that this function gets an integer and a float from the user
// The '&' passes the variables by reference instead of value as in previous examples
void getIntNFloatInput(int *anInt, float *aFloat){
  // Get integer from user
  printf("Enter an integer: ");
  scanf("%d", anInt);   // Note that there is no & because anInt is a reference
  // Get float from user
  printf("Enter an float: ");
  scanf("%f", aFloat);
  printf("In function %d %f\n", *anInt, *aFloat);
  // Return to calling function
  return;
}

// Get integer input from user
int getIntBetween1And10(int *anInt){
  *anInt = 0;
  // Loop will iterate until an int between 1 and 10
  // inclusive is entered
  while((*anInt < 1) || (*anInt > 10)){
    // Get integer from user
    printf("Enter an integer between 1 and 10: ");
    scanf("%d", anInt);
  }
  // Check that int is between 1 and 10
  if((*anInt >= 1) && (*anInt <= 10))
    return 1;
  else
    return 0;
}
```

▼ Ch03LecCodeWashers.c

```
/*
  This program calculates the weight of a number of
  washers.  The number and dimensions of washers are
  entered at the command line by the user.
  Inputs:
    density of material
    number of washers
    diameter of washer
    diameter of hole
    height of washer
  Output
    total weight of washers
*/

// Preprocessor directives
#include <stdio.h>
#include <math.h>
#define PI 3.14159

// Function prototypes - declare functions that are defined after
// the main function
void getInput(double*, double*, double*, double*, double*);
void setInput(double*, double*, double*, double*, double*);
double calcTotalWeight(double, double, double, double, double);
double calcWeightWasher(double, double, double, double);
double calcVolumeWasher(double, double, double);
```

```
double calcAreaWasher(double, double);
double calcAreaCircle(double);

// Driver functions to test other functions
void driver();
void driver2(double, double, double, double, double);
// Function to set variables for testing
void setInput(double*, double*, double*, double*, double*);

// Main function - entry point for all C programs
int main(){

  // Declare variables for inputs
  double density=0.0, numWashers=0.0, diaWasher=0.0, diaHole=0.0, height=0.0;
  // Declare variable for output
  double totalWeight = 0.0;

  // Call driver function to test functions
  //driver();

  // To set variables for faster testing
  //setInput(&density, &numWashers, &diaWasher, &diaHole, &height);

    // Call driver function to test functions with input variables
  //driver2(density, numWashers, diaWasher, diaHole, height);

  // Code to call functions to calculate the
  // Get input
  getInput(&density, &numWashers, &diaWasher, &diaHole, &height);
  printf("The inputs are: %1.2f %1.2f %1.2f %1.2f %1.2f\n",
    density, numWashers, diaWasher, diaHole, height);
  // Calc weight
  totalWeight = calcTotalWeight(density, numWashers, diaWasher, diaHole, height);
  // Print output
  printf("The total weight if %1.0f is %1.3f", numWashers, totalWeight);

  return 0;
}


// Get input from user
void getInput(double *density, double *numWashers, double *diaWasher, double *diaHole, double *height){
  printf("Enter density: ");
  scanf("%lf", density);
  printf("Enter number of washers: ");
  scanf("%lf", numWashers);
  printf("Enter diameter of washer: ");
  scanf("%lf", diaWasher);\
  printf("Enter diameter of hole: ");
  scanf("%lf", diaHole);
  printf("Enter height if washer: ");
  scanf("%lf", height);
  return;
}

// Calculate the total weight of a number of washers
double calcTotalWeight(double density, double numWashers, double diaWasher, double diaHole, double height){
  // Return total weight of washers
  return numWashers * calcWeightWasher(density, diaWasher, diaHole, height);
}

// Calculate the weight of a washer
double calcWeightWasher(double density, double diaWasher, double diaHole, double height){
  // Return the weight of a washer
  return density * calcVolumeWasher(diaWasher, diaHole, height);
}

// Calculate the volume of a washer
double calcVolumeWasher(double diaWasher, double diaHole, double height){
  // Return the volume of a washer
  return height * calcAreaWasher(diaWasher, diaHole);
}
```

```
// Calculate the area of a washer
double calcAreaWasher(double diaWasher, double diaHole){
  // Return the area of a washer
  return calcAreaCircle(diaWasher) - calcAreaCircle(diaHole);
}

// Calculate the area of a circle
double calcAreaCircle(double diameter){
  // Return the area of a circle
  return (PI * pow((diameter/2),2));
}

// Driver function to test calculations
void driver(){

  // Test area of a circle - answer should be 3.14
  printf("Area of a 2.0 circle: %1.2f\n", calcAreaCircle(2.0));
  // Test area of a circle - answer should be 0.2
  printf("Area of a 0.5 circle: %1.2f\n", calcAreaCircle(0.5));
  // Test area of a washer - answer should be 2.35
  printf("Area of a 2.0 washer with a 0.5 hole: %1.2f\n", calcAreaWasher(2.0, 0.5));
  // Test volume of a washer - answer should be 0.58
  printf("Volume of a 2.0 washer with a 0.5 hole and a 0.2 height: %1.2f\n",
    calcVolumeWasher(2.0, 0.5, 0.2));
  // Test density of a washer - answer should be 4.65
  printf("Density of a 2.0 washer with a 0.5 hole, a 0.2 height and density 7.9: %1.2f\n",
    calcWeightWasher(7.9, 2.0, 0.5, 0.2));
  // Test total weight of washers -  answer should be 23.27
  printf("Total weight of 5 washers with 2.0 washer diameter, 0.5 diameter hole, 0.2 height and 7.9 density: %1.2f\n",
    calcTotalWeight(7.9, 5.0, 2.0, 0.5, 0.2));
  return;
}

// Driver function to test calculations
void driver2(double density, double numWashers, double diaWasher, double diaHole, double height){

  // Test area of a circle
  printf("Area of a 2.0 circle: %1.2f\n",
    calcAreaCircle(diaWasher));
  // Test area of a circle
  printf("Area of a 0.5 circle: %1.2f\n",
    calcAreaCircle(diaHole));
  // Test area of a washer
  printf("Area of a 2.0 washer with a 0.5 hole: %1.2f\n",
    calcAreaWasher(diaWasher, diaHole));
  // Test volume of a washer
  printf("Volume of a 2.0 washer with a 0.5 hole and a 0.2 height: %1.2f\n",
    calcVolumeWasher(diaWasher, diaHole, height));
  // Test density of a washer
  printf("Density of a 2.0 washer with a 0.5 hole, a 0.2 height and density 7.9: %1.2f\n",
    calcWeightWasher(density, diaWasher, diaHole, height));
  // Test total weight of washers
  printf("Total weight of 5 washers with 2.0 washer diameter, 0.5 diameter hole, 0.2 height and 7.9 density: %1.2f\n",
    calcTotalWeight(density, numWashers, diaWasher, diaHole, height));
  return;
}

// Function to set values for testing and faster program execution
void setInput(double *density, double *numWashers, double *diaWasher, double *diaHole, double *height){
  *density = 7.9;
  *numWashers = 5;
  *diaWasher = 2.0;
  *diaHole = 0.5;
  *height = 0.2;
  return;
}
```

Chapter 4

If and Switch Statements

▼ Lecture PDF

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/fb435c38-56ca-4a86-8efa-4ac1181aab06/CIS1057Ch04.pdf

▼ Conditions

- Contain variables, constants and relational operators
- Relational expressions are 1 if true and 0 if false
- English terms

TABLE 4.2 Sample Conditions

Operator	Condition	English Meaning	Value
`<=`	`x <= 0`	`x` less than or equal to 0	1 (true)
`<`	`power < MAX_POW`	`power` less than `MAX_POW`	0 (false)
`>=`	`x >= y`	`x` greater than or equal to `y`	0 (false)
`>`	`item > MIN_ITEM`	`item` greater than `MIN_ITEM`	1 (true)
`==`	`mom_or_dad == 'M'`	`mom_or_dad` equal to `'M'`	1 (true)
`!=`	`num != SENTINEL`	`num` not equal to `SENTINEL`	0 (false)

▼ Logical Operators

Logical Operators

- Logical expressions
- Used for compound relational expressions
- AND &&, OR || and NOT !
  - x < y && y > z
  - x < y || y > z
  - !(x < y)
  - !(any non-zero value or expression) = 0
  - !(0) = 1
- Exclusive OR

▼ **Operator Precedence**

- Arithmetic follows algebraic rules
- Add relational and Logical

TABLE 4.6 Operator Precedence

Operator	Precedence
function calls	highest
! + - & (unary operators)	↓
* / %	
+ -	
< <= >= >	
== !=	
&&	
||	
=	lowest

Comparing Floating-Point Numbers

```
int main(){
  double x = 10.0;
  double y = 10.000003;
  double e = 0.009;
  if(x == y)
  printf("They are equal");
  if(fabs(x-y) < e)
  printf("They are approximately equal");
  return 0;
}
```

Complementing a Condition

```
DeMorgan's Law – reverse all symbols in
expression
  – !(<) = (>=) and !(>) = (<=)
  – !(==) = (!=) and !(!=) = (==)
  – !(&&) = (||) and !(||) = (&&)
• Examples
  – !(x < y) = (x >= y)
  – !(x == y) = (x != y)
  – !(x < y && y > z) = (x >= y || y <= z)
  – !(x < y || y > z) = (x >= y && y <= z)
```

▼ The Switch Statement

- Similar to If-Else
- Switch on condition in data – Can use char, int or enum
- Case for each alternative
- Break stops execution of multiple cases

```
enum Colors {Red, Green, Blue}; …
Colors color = Green;
  switch(color){
    case Red:
      printf("Color is Red\n");
      break;
    case Green:
      printf("Color is Green\n");
      break;
    case Blue:
        printf("Color is Blue\n");
        break;
default:
  printf("Unknown color\n");
}
```

▼ Class Example Code

▼ Ch04LecCode.c

```
#include <stdio.h>

int main(){
```

```
// Begin Chapter 4.1 - Compound Statements ***********************************************
/*
  Compound statements group blocks of code containing statements and expressions into
  containers with there own scope of influence over data variables.  These compound
  statements are typically used to group multiple statements and expressions to
  facilitate their use in control or loop structures.
*/
// This j has scope inside and outside of compound statements
int j = 1;

// Compound statements can contain statements or expressions
{
  // This i has scope inside the compound statement but not outside
  int i = 0;
  printf("This is line %d inside of a compound statement. j = %d\n", ++i, j++);
  printf("This is line %d inside of a compound statement. j = %d\n", ++i, j++);
  printf("This is line %d inside of a compound statement. j = %d\n", ++i, j++);
}
// This i has scope outside the compound statement
int i = 0;
printf("This is line %d outside of a compound statement. j = %d\n", ++i, j++);
// End Chapter 4.1 ***********************************************************************



// Begin Chapter 4.2 - Conditions *****************************************************
/*
  Conditions are the expressions that are evaluated to determine whether or not code is
  executed in if statements and when loops are terminated.
*/
// Variables to compare in conditional expressions
double x = 15.0, y = 25.0;
// Variable to contain result of condition

/*
  Conditional statements use relational operators (<, >, <=, >=) and equality operators (==, !=)
  are used to ask questions about data to decide whether a condition is true (non-zero) or false
  (zero).
*/
// Is x not equal to y
i = x != y;
printf("%.1f != %.1f is %d\n", x, y, i);
// Is x less than x
i = x < x;
printf("%.1f < %.1f is %d\n", x, x, i);
// Is x greater than or equal to y - x
i = x >= y - x;
printf("%.1f >= %.1f - %.1f is %d\n", x, y, x, i);
// Is x equal to y + x - y
i = x == y + x - y;
printf("%.1f == %.1f + %.1f - %.1f is %d\n", x, y, x, y, i);

/*
  Logical operators AND (&&) and OR (||) are used to expand conditional statements to logical
  expressions.  The negation operator NOT (!) reverses the true or false decision for a conditional
  statement or logical expression.

  Operator precedence
  Highest   ! + - &
        * / %
        + -
        < <= > >=
        == !=
        &&
        ||
  Lowest    =
*/
// Variables to compare logical expressions
int a = 6, b = 9, c = 14, flag = 1;
// Is c equal to a + b OR not flag
i = c == a + b || !flag;
printf("%d == %d + %d || %d is %d\n", c, a, b, !flag, i);
```

```
// Is a not equal to 7 AND flag OR c greater than or equal to 6
i = a != 7 && flag || c >= 6;
printf("%d != 7 && %d || %d >= 6 is %d\n", a, flag, c, i);
// Is NOT (b greater than or equal to 12) AND the remainder of a/2 equal to 0
i = !(b <= 12) && a % 2 == 0;
printf("!(%d <= 12) && %d %% 2 == 0 is %d\n", b, a, i);
// Is NOT (b greater than or equal to 12) AND the remainder of a/2 equal to 0
i = !(a > 5 || c < a + b);
printf("!(%d > 5 || %d  < %d + %d) is %d\n", a, c, a, b, i);

// Exclusive OR
a = 1;
b = 1;
i = (a == 1 || b == 1) && !((a == 1) && (b == 1));
printf("(%d == 1 || %d == 1) && !((%d == 1) && (%d == 1)) is %d\n", a, b, a, b, i);
b = 0;
i = (a == 1 || b == 1) && !((a == 1) && (b == 1));
printf("(%d == 1 || %d == 1) && !((%d == 1) && (%d == 1)) is %d\n", a, b, a, b, i);

// End Chapter 4.2 ************************************************************************



// Begin Chapter 4.3 - The if Statement ************************************************
/*
  The if statement decides whether or not to execute certain code statements and expressions.
*/
a = 1;
b = 2;
c = 3;

// if with single relational statement
if(a < b)
  printf("%d is less than %d\n", a, b);

// if-else with logical AND
if(a < b && b > c)
  printf("%d is less than %d AND b is greater than c is true\n", a, b);
else
  printf("%d is less than %d AND %d is greater than %d is false\n", a, b, b, c);

// End Chapter 4.3 ************************************************************************



// Begin Chapter 4.4 - if Statements with Compound Statements ***************************
/*
  if statements can contain compound statements.
*/
a = 1;
b = 2;
c = 3;

// if statement with one relational statement but 3 statements in a compound statement
if(a < b){
  printf("%d is less than %d\n", a, b);
  printf("%d is less than %d\n", a, b);
  printf("%d is less than %d\n", a, b);
}

// if-else with logical AND and compound statements
if(a < b && b > c){
  printf("%d is less than %d AND b is greater than c is true\n", a, b);
  printf("%d is less than %d AND b is greater than c is true\n", a, b);
  printf("%d is less than %d AND b is greater than c is true\n", a, b);
}
else{
  printf("%d is less than %d AND %d is greater than %d is false\n", a, b, b, c);
  printf("%d is less than %d AND %d is greater than %d is false\n", a, b, b, c);
  printf("%d is less than %d AND %d is greater than %d is false\n", a, b, b, c);
}

// End Chapter 4.4 ************************************************************************
```

```
// Begin Chapter 4.5 - Decision Steps in Algorithms ************************************
/*

*/
// End Chapter 4.5 ***************************************************************



// Begin Chapter 4.6 - More Problem Solving ********************************************
/*

*/
// End Chapter 4.6 ***************************************************************



// Begin Chapter 4.7 - Nested if Statements and Multiple-Alternative Decisions ************
/*
  switch-case statements can also select from a group of multiple alternatives.
*/
a = 1;
b = 2;
c = 3;

//
if(a < b)
  if(b < c)
    printf("%d is less than %d AND %d is greater than %d is true\n", a, b, b, c);
  else
    printf("%d is less than %d AND %d is greater than %d is false\n", a, b, b, c);

//
if(a < b){
  printf("%d is less than %d\n", a, b);
  if(b < c)
    printf("%d is less than %d AND %d is greater than %d is true\n", a, b, b, c);
}
else
    printf("%d is not less than %d\n", a, b);

//
if(a < 1)
  printf("%d is less than 1\n", a);
else if(a < 5)
  printf("%d is less than 5\n", a);
else if(a < 10)
  printf("%d is less than 10\n", a);
else
  printf("%d is not less than 10\n", a);

//
if(a < 1){
  printf("%d is less than 1\n", a);
  printf("%d is less than 1\n", a);
}
else if(a < 5){
  printf("%d is less than 5\n", a);
  printf("%d is less than 5\n", a);
}
else if(a < 10){
  printf("%d is less than 10\n", a);
  printf("%d is less than 10\n", a);
}
else{
  printf("%d is not less than 10\n", a);
  printf("%d is not less than 10\n", a);
}

// End Chapter 4.7 ***************************************************************
```

```
  // Begin Chapter 4.8 - The switch Statement ***********************************************
  /*

  */
  a = 1;
  b = 2;
  c = 3;

  // switch with int
  switch (a){
    case 0:
      printf("a is 0: %d\n", a);
      break;
    case 1:
      printf("a is 1: %d\n", a);
      break;
    case 2:
      printf("a is 2: %d\n", a);
      break;
    default:
      printf("a is not in case: %d\n", a);
      break;
  }

  char d = 'd';

  // switch with char
  switch (d){
    case 'A':
    case 'a':
      printf("d is A or a: %c\n", d);
      break;
    case 'B':
    case 'b':
      printf("d is B or b: %c\n", d);
      break;
    case 'C':
    case 'c':
      printf("d is C or c: %c\n", d);
      break;
    case 'D':
    case 'd':
      printf("d is D or d: %c\n", d);
      break;
    default:
      printf("d is not in case: %c\n", d);
      break;
  }


  // End Chapter 4.8 ******************************************************************



  // Begin Chapter 4.9 - Common Programming Errors ******************************************
  /*

  */
  // End Chapter 4.9 ******************************************************************

  // Return control to OS
  return 0;
}
```

▼ Ch04LecCodeAccount.c

```
// This program calculates an account balance
// after adjusting for interest and penalties
#include <stdio.h>
```

```
// Constant data for interest and penalties
const double OVERDRAWN_PENALTY = 8.00;
const double INTEREST_RATE = 0.02;    //2% annually

int main(){
  // Balance of account
    double balance;

  // Get balance from user
    printf("Enter your checking account balance: $");
  scanf("%lf", &balance);
    printf("Original balance $%.2f\n", balance);

  // If balance > 0 pay interest else charge penalty
    if (balance >= 0)
        balance = balance + (INTEREST_RATE * balance) / 12;
    else
        balance = balance - OVERDRAWN_PENALTY;

  // Print resulting balance to user
    printf("After adjusting for one month of interest and penalties,\n");
    printf("your new balance is $%.2f\n", balance);

  // Return control to OS
  return 0;
}
```

▼ Ch04LecCodeBMI.c

```
// This program calculates a person's BMI
// given his/her weight and height
#include <stdio.h>

int main(){
  // Declare variables for standard measures
  int pounds, feet, inches;
  // Declare variables for metric measures and BMI
  double heightMeters, mass, BMI;

  // Get standard measures for weight and height data from user
  printf("Enter your weight in pounds: ");
  scanf("%d", &pounds);
  printf("Enter your height in feet followed \nby a space then additional inches: ");
  scanf("%d %d", &feet, &inches);

  // Convert standard measures to meters and kilograms
  heightMeters = ((feet * 12) + inches) * 0.0254;
  mass = (pounds / 2.2);

  // Compute and print BMI
  BMI = mass / (heightMeters * heightMeters);
  printf("Your BMI is %f\n", BMI);

  // Evaluate and print health category
  printf("Your risk category is: " );
  if (BMI < 18.5)
    printf("Underweight\n");
  else if (BMI < 25)
    printf("Normal weight\n");
  else if (BMI < 30)
    printf("Overweight\n");
  else
    printf("Obese\n");

  // Return control to OS
  return 0;
}
```

▼ Ch04LecCodeGrade.c

```
// This program determines a student's letter
// grade considering his/her numerical score
#include <stdio.h>

int main(){
  // Variables for score and grade
    double score;
    char grade;

  // Get score from user
    printf("Enter your score: ");
  scanf("%lf", &score);

  // Determine letter grade based on score
    if (score >= 90)
        grade = 'A';
    else if (score >= 80)
        grade = 'B';
    else if (score >= 70)
        grade = 'C';
    else if (score >= 60)
        grade = 'D';
    else
        grade = 'F';

  // Print score and grade to screen
    printf("Score = %f\n", score);
    printf("Grade = %c\n", grade);

  // Return control to OS
  return 0;
}
```

▼ Ch04LecCodeNumberBabies.c

```
// This program calculates a person's BMI
// given his/her weight and height
#include <stdio.h>

int main(){
  // Declare variable for number of babies
    int numberOfBabies;

  // Get number of babies from user
  do{
    printf("Enter number of babies: ");
    scanf("%d", &numberOfBabies);
  while((numberOfBabies < 1) && (numberOfBabies > 10));

  // Use switch-case to evaluete a response
    switch (numberOfBabies)
    {
        case 1:
            printf("Congratulations.\n");
            break;
        case 2:
            printf("Wow. Twins.\n");
            break;
        case 3:
            printf("Wow. Triplets.\n");
            break;
        case 4:
        case 5:
            printf("Unbelieveable! ");
      printf("%d babies.\n", numberOfBabies);
            break;
        default:
```

```
            printf("I don't believe you.\n");
            break;
    }

  // Return control to OS
  return 0;
}
}
```

Chapter 5

Iteration and Loops

▼ Lecture PDF

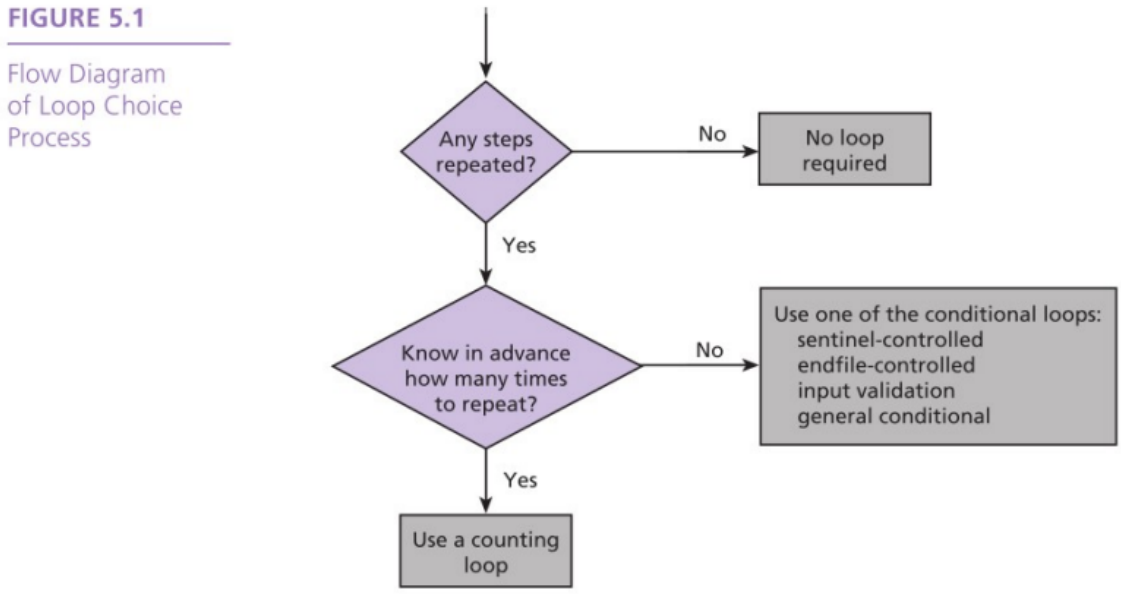
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0536f0ac-b08b-4375-a0c6-ee4cd01fe71d/CIS1057Ch05.pdf

▼ 🔁 Loop Types

- counting loop (while, for)
- sentinel-controlled loop (while, for, do-while)
- endfile-controlled loop (while, for, do-while)
- input validation loop (do-while)
- general conditional loop (while, for)

▼ Choosing a Loop Type

Choosing a Loop Type



🔁 Loop examples

For loop

```
for(int i=0; i<10; i++)
  printf("%d ", i)
```

Nested Loop

```
#include <stdio.h>
int main(){
// Loop iterators
int i = 0, j = 0;
// Print labels
  printf(" i j\n");
  for(i=0; i<4; i++){ // Loop through i
    printf("Outer %6d\n", i); // Print outer loop iterator
  for(j=0; j<i; j++) // Loop through j
    printf("Inner %9d\n", j); // Print inner loop iterator
  }
return 0;
}
```

While loop

```
int i = 0;
  while(i < 5)
    i++;
  printf(“i = %d\n”, i)

// exaple 2
int product = 1, item = 0;
  while(product < 10000){
    printf("%d\n", product);
    printf("Enter next item> ");
    scanf("%d", &item);
    product *= item;
}
```

do-while

```
int main () {

   /* local variable definition */
   int a = 10;

   /* do loop execution */
   do {
      printf("value of a: %d\n", a);
      a = a + 1;
   }while( a < 20 );

   return 0;
}
```

▼ array

▼ `int array`

```
#include <stdio.h>

int main() {
    //array 101
    int array[5];
    // value on the array
    array[0]=1;
    array[1]=2;
    array[2]=3;
    array[3]=4;
    array[4]=5;
    //print the value
    printf("%d\n",array[0]);
    printf("%d\n",array[1]);
    printf("%d\n",array[2]);
    //print the memory address
    printf("%d\n",&array[0]);
    printf("%d\n",&array[1]);
    printf("%d\n",&array[2]);

    //output
    1
    2
    3
    //memory address
    109680496
    109680500
    109680504
    // each memory is 4 byte

    return 0;
}
```

▼ `double array`

```
#include <stdio.h>

int main() {
    //array 101
    double array[5];
    // value on the array
    array[0]=123.5387;
    array[1]=234.4245;
    array[2]=345.4534;
    array[3]=456.6565;
    array[4]=567.3434;
    //print the value
    printf("%lf\n",array[0]);
    printf("%lf\n",array[1]);
    printf("%lf\n",array[2]);
    //print the memory address
    printf("%d\n",&array[0]);
    printf("%d\n",&array[1]);
    printf("%d\n",&array[2]);
    //each memory is 8 byte

    // output
    123.538700
    234.424500
    345.453400
    //memory address
    901311632
    901311640
    901311648



    return 0;
}
```

```
#include <stdio.h>
void create_array(double array[],int length);
void print_array(double array[],int length);
int main() {
    int length = 100;
    double array[length];
    create_array(array,length);
    print_array(array,length);

    return 0;
}

void create_array(double array[],int length){
    for (int i = 0; i <length;i++){
        array[i]=i+1.0;
    }
}
void print_array(double array[],int length){
    for (int i = 0; i <length;i++){
        printf("%.2f\n",array[i]);
    }
}
```

▼ `char array`

```
#include <stdio.h>

int main() {
    char array[]="hello";
```

```
    //print the whole string
    printf("%s\n",array);
    //print only one char
    printf("%c\n",array[1]);
    return 0;
}

//out put
hello
e
```

▼ `pointer array`

```
#include <stdio.h>
void create_array(double array[],int length);
void print_array(double array[],int length);
int main() {
    int length = 10;
    double array[length];
    double *ptr = array;
    for (int i=0;i<length;i++){
        array[i]=i+1.0;
        printf("array = %f %f\n",array[i],ptr[i]);
    }

    return 0;
}
// out put
array = 1.000000 1.000000
array = 2.000000 2.000000
array = 3.000000 3.000000
array = 4.000000 4.000000
array = 5.000000 5.000000
array = 6.000000 6.000000
array = 7.000000 7.000000
array = 8.000000 8.000000
array = 9.000000 9.000000
array = 10.000000 10.000000
```

```
//to review

// Online C compiler to run C program online
#include <stdio.h>
void create_array(double array[],int length);
void print_array(double array[],int length);
int main() {
    int length = 10;
    double array[length];
    /*
    double *ptr = array;
    for (int i=0;i<length;i++){
        array[i]=i+1.0;
        printf("array = %f %f\n",array[i],ptr[i]);
    }
    */
    printf("Using the index int to return value = %d"index_min(array,lenght));
    printf("Using the value int to return value = %f"value_min(array,lenght));

    return 0;
}
int index_min(double array[],int length){
    int min = 0;
    for (int i = 0; i <length;i++){
       if(array[i]<array[min]){
           min = i;
           min=array[i];
```

```
        }
    }
    return min;
}
double value_min(double array[],int length){
    int min = 0;
    for (int i = 0; i <length;i++){
       if(array[i]<array[min]){
           min=array[i];
       }
    }
    return min;
}


void create_array(double array[],int length){
    for (int i = 0; i <length;i++){
        array[i]=i+1.0;
    }
}


void print_array(double array[],int length){
    for (int i = 0; i <length;i++){
        printf("%.2f\n",array[i]);
    }
}
```

▼ String

```
// Print char array with loop
  printf("Print a char array with a loop\n");
  for(i=0; i<12; i++)
    printf("%c ", hello[i]);
  printf("\n\n");
  //hello[12] = 'a';
  // Print array as a string
  printf("Print a char array as a string\n");
  printf("%s\n\n", hello);

  // An array of strings is an array of pointers
  char month[12][10] = {"January", "February", "March", "April", "May", "June",
        "July", "August", "September", "October", "November", "December"};

  // Print months
  printf("Print months\n");
  for(i=0; i<12; i++)
    printf("%s\n", month[i]);
  printf("\n");
```

▼ struct

```
#include <stdio.h>
#include <string.h>

typedef struct{
  int id;
  int year;
  int miles;
  char make[50];
  char model[50];
  double price;
}car;
```

```
int main(){
  car c;
  c.id=123;
  strcpy(c.make,"f150");
  strcpy(c.model,"f150");
  c.year=2016;
  c.miles=10324;
  c.price=15000.0;

  printf("%d \n",c.id);
  printf("%s \n",c.make);
  printf("%s \n",c.model);
  printf("%d \n",c.year);
  printf("%d \n",c.miles);
  printf("$%.2f \n",c.price);


  return 0;
}
```

```
//using array
  car d = {1234,"Honda","civic",2016,78000,15500};
  typedef struct{
  int id;
  char make[50];
  char model[50];
  int year;
  int miles;
  double price;
}car;printf("%d \n",d.id);
  printf("%s \n",d.make);
  printf("%s \n",d.model);
  printf("%d \n",d.year);
  printf("%d \n",d.miles);
  printf("$%.2f \n",d.price);
  printf("\n");
```

```
//copying struct
car e = d;
  printf("%d \n",e.id);
  printf("%s \n",e.make);
  printf("%s \n",e.model);
  printf("%d \n",e.year);
  printf("%d \n",e.miles);
  printf("$%.2f \n",e.price);
```

```
//using print fuction

#include <stdio.h>
#include <string.h>

typedef struct{
  int id;
  char make[50];
  char model[50];
  int year;
  int miles;
  double price;
}car;
//prototype
void print_car(car);

int main(){
  car c;
  c.id=123;
  strcpy(c.make,"f150");
```

```
  strcpy(c.model,"f150");
  c.year=2016;
  c.miles=10324;
  c.price=15000.0;
  print_car(c);
  car d = {1234,"Honda","civic",2016,78000,15500};
  print_car(d);

return 0;
}
void print_car(car c){
  printf("%d \n",c.id);
  printf("%s \n",c.make);
  printf("%s \n",c.model);
  printf("%d \n",c.year);
  printf("%d \n",c.miles);
  printf("$%.2f \n",c.price);
  printf("\n");
}
```