

**CSE1301**  
**Computer Programming**  
**Lecture 11:**  
**Iteration (Part 2)**

1

- Topics**
- Infinite loops
  - **while** and **for**
  - Macros
  - Input from a file
  - Examples
    - CountConsonantsAndVowels
    - Factorization
- 2

**Infinite Loops**

```
while ( 1 )
{
  ...etc...etc...etc...
}

for ( ; 1 ; )
{
  ...etc...etc...etc...
}

for ( ; ; )
{
  ...etc...etc...etc...
}
```

3

**Infinite Loops**

```
while ( 1 )
{
  ...etc...etc...etc...
}

for ( ; 1 ; )
{
  ...etc...etc...etc...
}

for ( ; ; )
{
  ...etc...etc...etc...
}
```

Use an:

```
if ( condition )
{
  break;
}

statement to break the loop
```

4

**Example: asciiCheck.c**

```
while (1)
{
  printf("Enter bounds (low high): ");
  scanf("%d %d", &low, &high);

  if ((low >= 0) && (high <= 127) && (low < high))
  {
    break;
  }
  else
  {
    printf("Bad bounds. Try again.\n");
  }
}
```

5

**Example: asciiCheck.c**

```
while (1)
{
  printf("Enter bounds (low high): ");
  scanf("%d %d", &low, &high);

  if ((low >= 0) && (high <= 127) && (low < high))
  {
    break;
  }
  else
  {
    printf("Bad bounds. Try again.\n");
  }
}
```

6

### while and for

A **for** loop can always be rewritten as an equivalent **while** loop, and vice-versa

7

### Example: asciiPrint

Print out a section of the ASCII table

for each character from the lower bound to higher bound  
 {  
   print its ascii value and ascii character  
 }

```

for ( ch = low; ch <= high; ch++ )
{
    printf("%d: %c\n", ch, ch);
}
    
```

`asciiPrint1.c`

```

ch = low;
while ( ch <= high )
{
    printf("%d: %c\n", ch, ch);
    ch++;
}
    
```

`asciiPrint2.c`

8

### Example: asciiPrint (cont)

```

for ( ch = low; ch <= high; ch++ )
{
    printf("%d: %c\n", ch, ch);
}
ch = low;
while (1)
{
    printf("%d: %c\n", ch, ch);
    if (ch < high)
    {
        ch++;
    }
    else
    {
        break;
    }
}
    
```

`asciiPrint3.c`

9

### Example: asciiPrint (cont)

```

for ( ch = low; ch <= high; ch++ )
{
    printf("%d: %c\n", ch, ch);
}

ch = low;
for (;;)
{
    printf("%d: %c\n", ch, ch);
    ch++;
    if (ch > high)
    {
        break;
    }
}
    
```

`asciiPrint4.c`

10

### Example: ascii1.c

```

#include <stdio.h>
/* Print a section of the ASCII table */

#define MIN 0
#define MAX 127

int main()
{
    int low, high;
    char ch;

    while (1)
    {
        printf("Enter bounds (low high): ");
        scanf("%d %d", &low, &high);

        if ((low >= MIN) && (high <= MAX) && (low < high))
        {
            break;
        }
        else
        {
            printf("Bad bounds. Retry.\n");
        }
    }

    for (ch=low; ch <= high; ch++)
    {
        printf("%d: %c\n", ch, ch);
    }

    return 0;
}
    
```

11

### Example: ascii1.c (cont)

```

#include <stdio.h>
/* Print a section of the ASCII table */

#define MIN 0
#define MAX 127

int main()
{
    int low, high;
    char ch;

    while (1)
    {
        printf("Enter bounds (low high):");
        scanf("%d %d", &low, &high);

        if ((low >= MIN) && (high <= MAX) && (low < high))
        {
            break;
        }
        else
        {
            printf("Bad bounds. Retry.\n");
        }
    }

    for (ch=low; ch <= high; ch++)
    {
        printf("%d: %c\n", ch, ch);
    }

    return 0;
}
    
```

12

**Example: ascii1.c (cont)**

```

#include <stdio.h>
/* Print a section of
the ASCII table */
#define MIN 0
#define MAX 127

int main()
{
    int low, high;
    char ch;

    while (1)
    {
        printf("Enter bounds (low high):");
        scanf("%d %d", &low, &high);

        if ((low >= MIN) && (high <= MAX)
            && (low < high))
        {
            break;
        }
        else
        {
            printf("Bad bounds. Retry.\n");
        }
    }
}

```

**Macro definition:**  
**#define identifier tokens**  
 All subsequent instances of **identifier** are replaced with its **tokens**

**Example 1: CountConsonantsAndVowels**

- Count the number of consonants and the number of vowels in a file
- Non-alphabetic characters should not be counted

14

**Algorithm**

```

open file for input
set consonantCount to 0
set vowelCount to 0
loop
{
    input ch
    if (end of file)
    {
        exit loop
    }

    if (ch is a vowel)
    {
        increment vowelCount
    }
    else if (ch is a consonant)
    {
        increment consonantCount
    }
}
close file
output consonantCount, vowelCount

```

15

**Algorithm (cont)**

```

open file for input
set consonantCount to 0
set vowelCount to 0
loop
{
    input ch
    if (end of file)
    {
        exit loop
    }

    if (ch is a vowel)
    {
        increment vowelCount
    }
    else if (ch is a consonant)
    {
        increment consonantCount
    }
}
close file
output consonantCount, vowelCount

```

16

**Algorithm (cont)**

```

open file for input
set consonantCount to 0
set vowelCount to 0
loop
{
    input ch
    if (end of file)
    {
        exit loop
    }

    if (ch is a vowel)
    {
        increment vowelCount
    }
    else if (ch is a consonant)
    {
        increment consonantCount
    }
}
close file
output consonantCount, vowelCount

```

17

**Algorithm (cont)**

```

open file for input
set consonantCount to 0
set vowelCount to 0
loop
{
    input ch
    if (end of file)
    {
        exit loop
    }

    if (ch is a vowel)
    {
        increment vowelCount
    }
    else if (ch is a consonant)
    {
        increment consonantCount
    }
}
close file
output consonantCount, vowelCount

```

18

**Algorithm (cont)**

```

open file for input
set consonantCount to 0
set vowelCount to 0
loop
{
  input ch
  if (end of file)
  {
    exit loop
  }

  if (ch is a vowel)
  {
    increment vowelCount
  }
  else if (ch is a consonant)
  {
    increment consonantCount
  }
}
close file
output consonantCount, vowelCount
    
```

19

**Algorithm (cont)**

```

open file for input
set consonantCount to 0
set vowelCount to 0
loop
{
  input ch
  if (end of file)
  {
    exit loop
  }

  if (ch is a vowel)
  {
    increment vowelCount
  }
  else if (ch is a consonant)
  {
    increment consonantCount
  }
}
close file
output consonantCount, vowelCount
    
```

20

**Algorithm (cont)**


```

open file for input
set consonantCount to 0
set vowelCount to 0
loop
{
  input ch
  if (end of file)
  {
    exit loop
  }

  if (ch is a vowel)
  {
    increment vowelCount
  }
  else if (ch is a consonant)
  {
    increment consonantCount
  }
}
close file
output consonantCount, vowelCount
    
```

21

*Read input from file?*



**Algorithm (cont)**


```

open file for input
set consonantCount to 0
set vowelCount to 0
loop
{
  input ch
  if (end of input)
  {
    exit loop
  }

  if (ch is a vowel)
  {
    increment vowelCount
  }
  else if (ch is a consonant)
  {
    increment consonantCount
  }
}
close file
output consonantCount, vowelCount
    
```

22

*First, implement with input from stdin stream. Once working, modify to get input from file*



**Program**

```

#include <stdio.h>
int main()
{
  int consonantCount, vowelCount ;
  char ch ;

  consonantCount = 0 ;
  vowelCount = 0 ;

  printf("\nInput has %d consonants and %d vowels.\n",
    consonantCount, vowelCount) ;

  return 0;
}
    
```

23

**Program**

```

consonantCount = 0 ;
vowelCount = 0 ;

/* For each character in the file in turn, test if it is a
consonant or a vowel, and adjust the total accordingly */
while ( scanf("%c", &ch) != EOF )
{
  
}

printf("\nInput has %d consonants and %d vowels.\n",
  consonantCount, vowelCount) ;
    
```

24

**Program**

```

/* For each character in the file in turn, test if it is
a consonant or vowel, and adjust total accordingly */
while ( scanf("%c", &ch) != EOF )
{
    if (ch == 'a' || ch == 'A' ||
        ch == 'e' || ch == 'E' ||
        ch == 'i' || ch == 'I' ||
        ch == 'o' || ch == 'O' ||
        ch == 'u' || ch == 'U')
    {
        /* Vowel */
        vowelCount++;
    }
    else if ((ch >= 'a' && ch <= 'z') ||
             (ch >= 'A' && ch <= 'Z'))
    {
        /* Consonant, since vowels already dealt with */
        consonantCount++;
    }
}
    
```

25

```

#include <stdio.h>
/* Count the number of vowels, and the number of consonants in the input */
int main()
{
    int consonantCount, vowelCount ;
    char ch ;
    consonantCount = 0 ;
    vowelCount = 0 ;

    /* For each character in the file in turn, test if it is
    a consonant or vowel, and if so, adjust total accordingly */
    while ( scanf("%c", &ch) != EOF )
    {
        if (ch == 'a' || ch == 'A' ||
            ch == 'e' || ch == 'E' ||
            ch == 'i' || ch == 'I' ||
            ch == 'o' || ch == 'O' ||
            ch == 'u' || ch == 'U')
        {
            /* Vowel */
            vowelCount++;
        }
        else if ( (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') )
        {
            /* Consonant, since vowels already dealt with. */
            consonantCount++;
        }
    }

    printf("\nInput has %d consonants and %d vowels.\n",
           consonantCount, vowelCount) ;
    return 0 ;
}
    
```

vowel1.c  
26

Modify to get input from file?

```

#include <stdio.h>
int main()
{
    int consonantCount, vowelCount ;
    char ch ;
    consonantCount = 0 ;
    vowelCount = 0 ;

    while ( scanf("%c", &ch) != EOF )
    {
        ...etc...etc...etc...
    }

    printf("\nFile has %d consonants and %d vowels.\n",
           consonantCount, vowelCount) ;

    return 0 ;
}
    
```

27

```

#include <stdio.h>
int main()
{
    FILE *inputFile ;
    int consonantCount, vowelCount ;
    char ch ;
    consonantCount = 0 ;
    vowelCount = 0 ;

    inputFile = fopen("yourFile.txt", "r") ;

    while ( fscanf(inputFile, "%c", &ch) != EOF )
    {
        ...etc...etc...etc...
    }

    printf("\nFile has %d consonants and %d vowels.\n",
           consonantCount, vowelCount) ;

    fclose(inputFile) ;
    return 0 ;
}
    
```

28

Input file stream  
(aka "file pointer")

```

#include <stdio.h>
int main()
{
    FILE *inputFile ;
    int consonantCount, vowelCount ;
    char ch ;
    consonantCount = 0 ;
    vowelCount = 0 ;

    inputFile = fopen("yourFile.txt", "r") ;

    while ( fscanf(inputFile, "%c", &ch) != EOF )
    {
        ...etc...etc...etc...
    }

    printf("\nFile has %d consonants and %d vowels.\n",
           consonantCount, vowelCount) ;

    fclose(inputFile) ;
    return 0 ;
}
    
```

29

Declare file pointer

Open file for input

Read input from file

Close file

```

#include <stdio.h>
/* Count the number of vowels, and the number of consonants in the file. */
int main()
{
    FILE *inputFile ;
    int consonantCount, vowelCount ;
    char ch ;
    inputFile = fopen("yourFile.txt", "r") ;
    consonantCount = 0 ;
    vowelCount = 0 ;

    /* For each character in the file in turn, test if it is
    a consonant or vowel, and if so, adjust total accordingly. */
    while ( fscanf(inputFile, "%c", &ch) != EOF )
    {
        if (ch == 'a' || ch == 'A' ||
            ch == 'e' || ch == 'E' ||
            ch == 'i' || ch == 'I' ||
            ch == 'o' || ch == 'O' ||
            ch == 'u' || ch == 'U')
        {
            /* Vowel */
            vowelCount++;
        }
        else if ( (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') )
        {
            /* Consonant, since vowels already dealt with */
            consonantCount++;
        }
    }

    printf("\nFile has %d consonants and %d vowels.\n",
           consonantCount, vowelCount) ;

    fclose(inputFile) ;
    return 0 ;
}
    
```

vowel2.c  
30

More on file input/output in Lecture 21

## Example 2: Factorization

- Write a program which prints out the prime factorization of a number (treat 2 as the first prime)
- For example, on input 6, desired output is: 2 3
  - " " 24, " " : 2 2 2 3
  - " " 14, " " : 2 7
  - " " 23, " " : 23 (*23 is prime*)

31

**Algorithm**

```
input n
set factor to 2
```

32

**Algorithm (cont)**

```
input n
set factor to 2
while(some factor yet to try)
{
```

33

**Algorithm (cont)**

```
input n
set factor to 2
while(some factor yet to try)
{
  if (n is divisible by factor)
  {
    output factor
    set n to n / factor
  }
}
```

34

**Algorithm (cont)**

```
input n
set factor to 2
while(some factor yet to try)
{
  if (n is divisible by factor)
  {
    output factor
    set n to n / factor
  }
  else
  {
    increment factor
  }
}
```

35

**Algorithm (cont)**

<pre>input n set factor to 2 while(some factor yet to try) {   if (n is divisible by factor)   {     output factor     set n to n / factor   }   else   {     increment factor   } }</pre>	<p><b>Why not?</b></p> <pre>while(some factor yet to try) {   if (n is divisible by factor)   {     output factor     set n to n / factor   }   increment factor }</pre>
--	--

36

```

#include <stdio.h>
/* Print out the prime factors of a number */
int main()
{
    //input n

    //set factor to 2

    //while some factor yet to try

    //if (n is divisible by factor)

    //output factor

    //set n to n/factor

    //increment factor

    return 0;
}

```

**factor1.c**  
37

```

#include <stdio.h>
/* Print out the prime factors of a number. */
int main()
{
    int n, factor;
    printf("Number integer: ");
    scanf("%d", &n);
    printf("The prime factors of %d are: ", n);
    /* Try each possible factor in turn. */
    for (factor = 2; factor <= n; )
    {
        if (n % factor == 0)
        {
            /* n is a multiple of factor,
            ** so print factor and divide n by factor. */
            printf("%d", factor);
            n = n / factor;
        }
        else
        {
            /* n is not a multiple of factor;
            ** try next possible factor. */
            factor++;
        }
    }
    printf("\n\n");
    return 0;
}

```

**factor2.c**

## Reading

- King
  - Chapter 6, except Section 6.2
  - Chapter 22, Sections 22.1-22.2
- Deitel and Deitel
  - Chapter 4