Introduction to Programming

Lecture Three

N.R.Aravind

I.I.T. Hyderabad

26 Sep 2017

Topics in this lecture

- Review
- Binary arithmetic
- **Arrays**
- while Loop examples

Review

Compiling and running a C program

$ gcc helloWorld.c -o hello

- gcc: Gnu C Compiler
- Translates the C program into machine code named “hello”
- -o: specifies the outpiut file name

$ ./hello

- Run (execute) the program named "hello"
- To run a file named "xyz", type ./xyz.

Lab guidelines

1. Write programs on paper before the lab.
2. Save programs (pen-drive, email, g-drive)
3. Meaningful program names:
   addition.c, circle.c, helloWorld.c, quiz.c, maximum.c
4. Avoid: abcd.c, ramakrishna.c, cs15btech1001.c, program1.c, progam2.c
5. Meaningful variable names:
   radius, area, count, username, country, length
6. Take breaks!

Lab reschedule

Next Mon lab $\Rightarrow$ This Wed, 12:00 and 4:00 p.m.

if (__) {...} else {...}

```
int a=5,b=15;
if (!(a > 7))
{
  printf(" Hello");
}
else
{
  printf(" Welcome");
}
if (!(b==15))
{
  printf(" Bye");
}
```

Exercise 3: printSquares.c

```
    int count, N, square;     // Accept the value of N.
    count=1;
    while (square<=N)
    {
      square=count*count;
      printf("\n %d",square);
      count=count+1;
    }
// Output statements
```

Exercise 3: printSquares.c

```
    int count, N;     // Accept the value of N.
    count=1;
    while (count*count<=N)
    {
      printf("\n %d", count*count);
      count=count+1;
    }
// Output statements
// Two multiplications
```

Exercise 3: printSquares.c

```
    int count, N, square;     // Accept the value of N.
    count=1;
    square=0;
    while (square<=N)
    {
      square=square+2*count-1;
      printf("\n %d",square);
      count=count+1;
    }
// Output statements
```

Useful basic block

```
i=1;
while(i<20)
{
// Some statements
i=i+1;
}
```

Useful basic block

```
sum=0;
i=1;
while(i<20)
{
sum=sum+i;
i=i+1;
}
```

Useful basic block

```
product=1;
i=1;
while(i<20)
{
product=product*i;
i=i+1;
}
```

Useful block for strings

```
i=0;
while(text[i]!='\0')
{
printf("\n %d",text[i]");
i=i+1;
}
```

Binary Arithmetic

Decimal

- $4716 = 4 \times 10^3 + 7 \times 10^2 + 1 \times 10^1 + 6 \times 10^0$.
- 4716=6+10+700+4000
- 583=3+80+500
- Decimal: Multiply by 1, 10, 100, 1000 etc. (right-to-left)
- In binary, we multiply by 1, 2, 4, 8 etc. (right-to-left)

Binary to Decimal

- Multiply by 1, 2, 4, 8 etc. (right-to-left)
- $(1101)_2 = 1 + 4 + 8 = 13.$
- $(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0.$
- In binary, $(100)_2 = 4$ and $(111)_2 = 7$.

Binary to decimal

- $101_2 =$

Binary to decimal

- $101_2 = 5$.

Binary to decimal

- $101_2 = 5$.
- $1000_2 =$

Binary to decimal

- $101_2 = 5$.
- $1000_2 = 8$.

Binary to decimal

- $101_2 = 5$.
- $1000_2 = 8$.
- $1110_2 =$

Binary to decimal

- $101_2 = 5$.
- $1000_2 = 8$.
- $1110_2 = 14$.

Binary to decimal

- $101_2 = 5$.
- $1000_2 = 8$.
- $1110_2 = 14$.
- $10101_2 =$

Binary to decimal

- $101_2 = 5$.
- $1000_2 = 8$.
- $1110_2 = 14$.
- $10101_2 = 21$.

Digits: right-to-left

- $4716 = 10 \times 471 + 6$
- $471 = 10 \times 47 + 1$
- $47 = 10 \times 4 + 7$
- $4 = 10 \times 0 + 4$.

Digits: right-to-left

- $4716 = 10 \times 471 + 6$
- $471 = 10 \times 47 + 1$
- $47 = 10 \times 4 + 7$
- $4 = 10 \times 0 + 4$.

- $14 = 2 \times 7 + 0$
- $7 = 2 \times 3 + 1$
- $3 = 2 \times 1 + 1$
- $1 = 2 \times 0 + 1$.

Printing the digits from right-to-left

```
int num;
int dividend, remainder;
dividend=num;
while (???)
{
    remainder=dividend % 10;
    // Update dividend.
}
```

char variable in memory

char ch='A';

Address	Value
68400	'A'

68400	0	1	0	0	0	0	0	1

// ASCII value of 'A' is $65 = (100001)_2$.

int variable in memory

int num=21;

Address	Value
68400	21

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	1	0	1

int variable in memory

int num=-21;

Address	Value
68400	-21

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	1	0	1

Arrays

Arrays

```
int num[20];
```

Arrays

int num[20];

Variable	Address	Value
num[0]	68400	
num[1]	68404	
num[2]	68408	
num[3]	68412	
num[18]	68472	
num[19]	68476	

Arrays

int num[20];

Variable	Address	Value
num[0]	68400	
num[1]	68404	
num[2]	68408	
num[3]	68412	
num[18]	68472	
num[19]	68476	

num[0]=287;
num[3]=-50;

Arrays

int num[20];

Variable	Address	Value
num[0]	68400	
num[1]	68404	
num[2]	68408	
num[3]	68412	
num[18]	68472	
num[19]	68476	

num[0]=287;
num[3]=-50;
x=2;

Arrays

int num[20];

Variable	Address	Value
num[0]	68400	
num[1]	68404	
num[2]	68408	
num[3]	68412	
num[18]	68472	
num[19]	68476	

num[0]=287;
num[3]=-50;
x=2;

num[x]=841;

Arrays

int num[100];

Variable	Address	Value
num[0]	68400	287
num[1]	68401	
num[2]	68402	841
num[3]	68403	-50
num[98]	68498	
num[99]	68499	

num[0]=287;
num[3]=-50;
x=2;

num[x]=841;

Integer arrays

```
int num[10], i=0;
while(i<10)
{
printf("Enter number %d",i);
scanf("%d",&num[i]);
i=i+1;
}
```

Using arrays

```
    int days[12]=
{0,31,59,80,110,141,171,202,233,263,294,324};

    // Add days[m] to calculate day-of-week
```

Using arrays

```
int days[12]=
{0,31,59,80,110,141,171,202,233,263,294,324};

// Add days[m] to calculate day-of-week

int matrix[10][10];

char days[7][3]={"Sun","Mon",...,"Sat"};
```

Examples

P1: Finding the maximum of a sequence of numbers

37

P1: Finding the maximum of a sequence of numbers

24

P1: Finding the maximum of a sequence of numbers

80

P1: Finding the maximum of a sequence of numbers

64

P1: Finding the maximum of a sequence of numbers

32

P1: Finding the maximum of a sequence of numbers

56

P2: Count the vowels in a string

Input: ALPHANUMERIC

Output: There are 5 vowels.

P3: Print logarithms

P4: Trigonometric ratios