FIT ICT Software Development

Lecture 6

Performing Operations

Lecturer : Charles Tyner

FIT ICT Software Development

Doing Arithmetic

The arithmetic operators used in C are identical to those familiar from Java

Operation	Operator
addition	+
subtraction	-
multiplication	*
division	/
modulus	%
increment	++
decrement	--

FIT ICT Software Development

Doing Arithmetic

The arithmetic operators used in C are identical to those familiar from Java

Precedence – the order of operations – is also the same

For greater control and readability, use brackets

Compare:

```
a = b*c-d%e/f
a = (b*c) – ((d%e)/f)
```

Increment and decrement can be placed before or after the operand for different results

FIT ICT Software Development

Doing Arithmetic

1. Begin a new program with a preprocessor instruction to include standard input/output library:

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

2. Add main function that declares and initialises several integer variables:

```
int main() {
        int a = 4, b = 8, c = 1, d = 1 ;
}
```

FIT ICT Software Development

Doing Arithmetic

3. In main function block, output results of operations performed on variable values:

```
printf("Addition: %d\n", a + b);
printf("Subtraction: %d\n", b - a);
printf("Multiplication: %d\n", a * b);
printf("Division: %d\n", b/a);
printf("Modulus: %d\n", a % b);
```

FIT ICT Software Development

Doing Arithmetic

4. Now in main function block, output results of both ***postfix*** and ***prefix*** operations :

```
printf("Postfix increment: %d\n", c++);
printf("Postfix now: %d\n", c);
printf("Prefix increment: %d\n", ++d);
printf("Prefix now: %d\n", d);
```

5. At end of function block return a zero integer

```
return 0;
```

6. Save, compile, execute, study

FIT ICT Software Development

Assigning Values

Shorthand forms of operations are available in C

Operator	Example	Equivalent
=	a = b	a = b
+=	a += b	a = (a+b)
-=	a -= b	a = (a-b)
*=	a *= b	a = (a*b)
/=	a /= b	a = (a/b)
%=	a %= b	a = (a%b)

Note that "=" means "assign" rather than "equals", which is represented by equality operator ==

FIT ICT Software Development

Assigning Values

1. Begin a new program with a preprocessor instruction to include standard input/output library:

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

2. Add main function that declares a constant value approximating mathematical value of *Pi*:

```
int main() {
        int a, b ;
}
```

FIT ICT Software Development

Assigning Values

3. In main function block, output results of operations performed on variable values:

```
printf("Assigned: \n");
printf("\tVariable a= %d\n", a = 8);
printf("\tVariable b= %d\n", b = 4);
printf("Added & assigned: \n");
printf("\tVariable a+=b (8+=4) a = %d\n", a += b);
printf("Subtracted & assigned: \n");
printf("\tVariable a-=b (12 -=4) a = %d\n", a += b);
```

FIT ICT Software Development

Assigning Values

```
printf("Multiplied & assigned: \n");
printf("\tVariable a*=b (8*=4) a = %d\n", a *= b);
printf("Divided & assigned: \n");
printf("\tVariable a/=b (32 /=4) a = %d\n", a
/= b);
printf("Modulated & assigned: \n");
printf("\tVariable a%%=b (8%%=4) a = %d\n", a
%= b);
```

4. Return 0. Save, compile, run, study output

FIT ICT Software Development

Comparing Values

The operators used in C to compare two numerical values are listed below

Operation	Operator
Equality	==
Inequality	!=
Greater than	>
Less than	<
Greater than or equal to	>=
Less than or equal to	<=

Operation returns an integer value: 1 for True and 0 for False

FIT ICT Software Development

Comparing Values

1. Begin a new program with a preprocessor instruction to include standard input/output library:

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

2. Add main function that declares and initialises three integer variables and two character variables:

```
int main() {
        int zero = 0, nil = 0, one = 1;
        char upr = 'A', lwr = 'a';
}
```

FIT ICT Software Development

Comparing Values

3. In main function block, output results of comparison operations performed on variables:

```
printf("Equality (0==0): %d\n", zero == nil);
printf("Equality (0==1): %d\n", zero == one);
printf("Equality (A==a): %d\n", upr == lwr);
printf("Inequality (A!=a): %d\n", upr != lwr);
printf("Greater than (1 > 0): %d\n", one > nil);
printf("Less than (1 < 0): %d\n", one < nil);
printf("Greater or equal (0 >= 0): %d\n", zero >= nil);
printf("Less or equal (1 <= 0): %d\n", one < =nil);
```

4. Return 0. Save, compile, run, study output

FIT ICT Software Development

Assessing Logic

The logical operators used in C are listed below

Operation	Operator
Logical AND	&&
Logical OR	||
Logical NOT	!

Operation returns a boolean value: 1 for True and 0 for False

Logical && AND returns True only if both operands being evaluated are themselves true; logical || OR is True if either operand is true

FIT ICT Software Development

Assessing Logic

1. Begin a new program with a preprocessor instruction to include standard input/output library:

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

2. Add main function that declares and initialises two integer variables:

```
int main() {
        int yes= 1, no = 0;
}
```

FIT ICT Software Development

Assessing Logic

3. In main function block, output results of logic operations performed on variables:

```
printf("AND (no&&no): %d\n", no && no);
printf("AND (yes&&no): %d\n", yes && no);
printf("AND (yes && yes): %d\n", yes && yes);
printf("OR (no||no): %d\n", no || no);
printf("OR (yes||no): %d\n", yes || no);
printf("OR (yes||yes): %d\n", yes || yes);
printf("NOT (yes !yes): %d%d\n", yes , !yes);
printf("NOT (no !no): %d%d\n", no, !no);
```

4. Return 0. Save, compile, run, study output

FIT ICT Software Development

Examining Conditions

The "ternary" operator ? Is a terse alternative to an IF statement (covered in next lecture)

It evaluates an expression for a true or false Boolean value, then executes one of two alternative statements depending on the result

Syntax:

(test-expression) ? if-true-do-this: if-false-do-this;

Could be used to check if a number is odd or even

(7 % 2 != 0) ? printf("Odd number"): printf("Even number");

FIT ICT Software Development

Examining Conditions

Also useful to ensure correct grammar in output regarding singular and plural items

Evaluation can be made within *printf()* statement

printf("There %s %d", (num == 1)? "is":"are", num);

If true that the value of num is one, then singular "is" will be used; otherwise the plural "are"

Can also be used to assign appropriate value to a variable depending on result of evaluation:

variable = (test-expression)? if-true-assign-this: if-false-assign-this;

FIT ICT Software Development

Examining Conditions

1. Begin a new program in the usual way:

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

2. Add main function that declares and initialises a single integer variable:

```
int main() {
        int num= 7;
}
```

FIT ICT Software Development

Examining Conditions

3. Next, add a conditional statement to output variable value's parity:

```
(num %2 != 0)?
printf("%d is odd\n", num) : printf("%d is even\n", num);
```

4. Now add conditional statement to check correct grammar for output:

```
printf("There %s",  (num == 1) ? "is": "are");
printf("%d %s\n", num, (num == 1) ?  "apple": "apples");
```

FIT ICT Software Development

Examining Conditions

5. Change the value of the variable to one and repeat the grammar check:

```
num = 1;
printf("There %s",  (num == 1) ? "is": "are");
printf("%d %s\n", num, (num == 1) ?  "apple": "apples");
```

4. Return 0. Save, compile, run, study output