CENG 230

Introduction to C Programming

Week 4 – Overview of C

Sinan Kalkan

Some slides/content are borrowed from Tansel Dokeroglu, Nihan Kesim Cicekli.

Previously on CEng 230!

Algebraic equality or relational operator	C equality or relational operator	Example of C condition	Meaning of C condition
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y

Fig. 2.12 | Equality and relational operators.

! exclamation mark

= is assignment and == is an equality operator

Increment, Decrement Operators

Previously on CEng 230!

Operator	Sample expression	Explanation
++	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

Fig. 3.12 | Increment and decrement operators

- ++a, --a

  vs

- a++, a--

Previously on CEng 230!

Assignment operators

=

+=

-=

*=

/=

%=

a+=10; is the same with

a=a + 10;

Compound Assignment Operators

Previously on CENG 230!

- var op= expr
- += -= *= /= %=

Some examples

Previously on CEng 230!

- i += j = k;
- i = j += k;

Relational Operators

Previously on CEng 230!

- < <= > >= == !=
- False means 0 (zero)
- True means anything that is not False (i.e., non-zero)

Operator	Type	Associativity
+ - ++ --	Unary	Right to left
* / %	Binary	Left to right
+ -	Binary	Left to right
< <= > >=	Binary	Left to right
== !=	Binary	Left to right
= *= /= %= += -=	Binary	Right to left

Example: a = b + c <= d + e == c - d

Today

- Finish up operators
- Type conversion
- Defining macros
- Examples
- Changing the flow of the program

Logical Operators

- && || !

Operator	Type	Associativity
+ - ++ -- !	Unary	Right to left
* / %	Binary	Left to right
+ -	Binary	Left to right
< <= > >=	Binary	Left to right
== !=	Binary	Left to right
&&	Binary	Left to right
||	Binary	Left to right
= *= /= %= += -=	Binary	Right to left

Type conversion (casting)

Type conversions (casting)

```
float a = 5.25;
int b = a;
/*Casting from float to int. The value of b here is 5*/


char c = 'A';
int x = c;
/*Casting from char to int.
The value of x here is 65: the ASCII code of 'A'*/

int x=7, y=5 ;
float z;
z=x/y;
/* the value of z is 1.00 */


int x=7, y=5;
float z;
z = (float)x/(float)y;
/ the value of z is 1.4*/
```

Type conversions (casting)

```
printf( "Welcome : %d", (3/2) );
```

Output is : 1 and **fraction** part of the number is lost

```
int sum = 17, count = 5;
double mean;
mean = (double) sum / count;
printf("Value of mean : %f\n", mean );
```

Value of mean : 3.400000

```
int i = 17;
char c = 'c'; /* ascii value is 99 */
int sum;
sum = i + c;
printf("Value of sum : %d\n", sum );
```

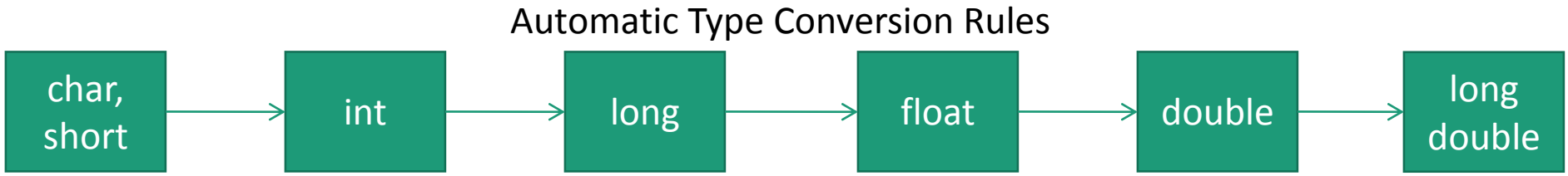
Value of sum : 116

TABLE 2.7 ASCII Codes for Characters

Character	ASCII Code
' '	32
'*'	42
'A'	65
'B'	66
'Z'	90
'a'	97
'b'	98
'z'	122
'0'	48
'9'	57

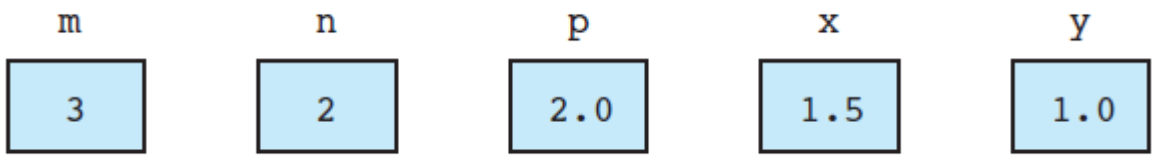
What is the result of printf("%d", ‘d’– ‘a’);

Type Conversion



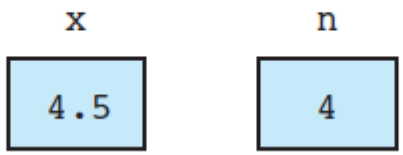
* Advice: Avoid automatic type conversion!

```
int  m, n;

double  p , x , y;

 m = 3;
 n = 2;
 p = 2.0;
 x = m / p;    /*  3/2.0  */
 y = m / n;    /*   3/2   */
```



```
x = 9 * 0.5;
n = 9 * 0.5;
```

evaluates to the real number `4.5`. If `x` is of type `double`, the number `4.5` is stored in `x`, as expected. If `n` is of type `int`, only the integral part of the expression value is stored in `n`, as shown.



Changing the flow of the program

If statements

Changing the flow of the program

- if statements

if(expr)
{ ….
}
else if(expr)
{…
}
…
else
{ … }

```
if(a > b)
        printf(“a is bigger”);
else if(a < b)
        printf(“b is bigger”);
else
        printf(“a = b”);
```

Changing the flow of the program

- Common mistake with if statements
- **if**(a = 10) { … }
- **if**(a == 10); { … }

Nested if statements

- **if**(…)

  if(…)

  {….}

  else

  {….}