CS11001/CS11002
Programming and Data Structures (PDS)

(Theory: 3-1-0)

Continuing with Data types…

Input Output for short and long int

```
#include<stdio.h>
void main( )
{
  short int shorti;
  long int longi;
 printf(“Input short int: “);
 scanf(“%hd”,&shorti);
 printf(“%hd\n”,shorti);
 printf(“Inputt long int: “);
 scanf(“%ld”,&longi);
 printf(“%ld\n”,longi);
 printf(“shorti = %hd and longi=%ld”,shorti,longi);
}
```

A Sample Run:

```
Input short int: 20
Input long int: 2000000
shorti= 20 and longi= 2000000
```

limits.h

- The header file limits.h contains the largest and smallest values that each of the various data types can hold.
- Typical path: /usr/lib
- The header file float.h contains the range of the double data type.
- Have a look at these files!

The **typedef** statement

- This statement can be used to define new data types.
  - For example:

    typedef unsigned long ulong;
    - ulong is a new data type equivalent to unsigned long
  - It can be used as any other data type as follows;

    ulong u;

    (declares u to be of the type ulong)
  - The size of the new data type can also be found in bytes using **sizeof(ulong)**

Pointer Data Type

- Pointers are addresses in memory.
- In order that the user can directly manipulate memory addresses, C provides an abstraction of addresses.
- The memory location where a data item resides can be accessed by a pointer to that particular data type. C uses the special character * to declare pointer data types.
- A pointer to a double data is of data type double *.
- A pointer to an unsigned long int data is of type unsigned long int *.
- A character pointer has the data type char *.

- We will study pointers more elaborately later in this course.

Examples

- int m, n, armadillo;
- int platypus;
- float hi, goodMorning;
- unsigned char _u_the_charcoal;

Constants

- Defining a data type is not enough.
- You need to assign the variables and work with specific values of various data types.
- Examples: PI (hopefully it will not change its value!)
- 1.0/n is our previous example of finding reciprocals has a constant.

Integer Constants

- An integer constant is a non-empty sequence of decimal numbers preceded optionally by a sign (+ or -).
- However, the common practice of using commas to separate groups of three (or five) digits is not allowed in C.
- Nor are spaces or any character other than numerals allowed.
- Here are some valid integer constants:

```
 332
  -3002
 +15
  -00001020304
```

- And here are some examples that C compilers do not accept:

```
3 332
2,334
-   456
2-34
12ab56cd
```

Hexadecimal values

- You can also express an integer in base 16, i.e., an integer in the **hexadecimal** (abbreviated **hex**) notation.
- In that case you must write either 0x or 0X before the integer. Hexadecimal representation requires 16 digits 0,1,...,15. In order to resolve ambiguities the digits 10,11,12,13,14,15 are respectively denoted by a,b,c,d,e,f (or by A,B,C,D,E,F). Here are some valid hexadecimal integer constants:

  0x12ab56cd -0X123456 0xABCD1234 +0XaBCd12

Constants of different Integer types

- Since different integer data types use different amounts of memory and represent different ranges of integers, it is often convenient to declare the intended data type explicitly.

Constants of different Integer types

Suffix	**Data type**
L (or l)	long
LL (or ll)	long long
U (or u)	unsigned
UL (or ul)	unsigned long
ULL (or ull)	unsigned long long

Examples

- 4000000000UL
- 123U
- 0x7FFFFFFFl
- 0x123456789abcdef0ULL

Real Constants

- Real constants can be specified by the usual notation comprising an optional sign, a decimal point and a sequence of digits. Like integers no other characters are allowed.
- Real numbers are sometimes written in the *scientific notation* (like 3.45×10^{67}). The following expressions are valid for writing a real number in this fashion: 3.45e67 +3.45e67 -3.45e-67 .00345e-32 1e-15.
- You can also use E in place of e in this notation

Character Constants

- Character constants are single printable symbols enclosed within single quotes.
- Here are some examples: 'A' '7' '@' ' '

Special Characters

Constant	Character	ASCII value
'\0'	Null	0
'\b'	Backspace	8
'\t'	Tab	9
'\n'	New line	13
'\''	Quote	39
'\\'	Backslash	92

Try this!

```
#include<stdio.h>
main()
{
  int i;
  for(i=0;i<10000;i++)
  prinf(“%c”,’\a’)
}
```

Formats

- %c The character format specifier.
  %d The integer format specifier.
  %i The integer format specifier (same as %d).
  %f The floating-point format specifier.
  %e The scientific notation format specifier.
  %E The scientific notation format specifier.
  %g Uses %f or %e, whichever result is shorter.
  %G Uses %f or %E, whichever result is shorter.
  %o The unsigned octal format specifier.
  %s The string format specifier.
  %u The unsigned integer format specifier.
  %x The unsigned hexadecimal format specifier.
  %X The unsigned hexadecimal format specifier.
  %p Displays the corresponding argument that is a pointer.
  %n Records the number of characters written so far.
  %% Outputs a percent sign.

Characters can be represented by numbers

- Since characters are identified with integers in the range -127 to 128 (or in the range 0 to 255), you can use integer constants in the prescribed range to denote characters.
- The particular sequence '\xuv' (synonymous with 0xuv) lets you write a character in the hex notation.
- For example, '\x2b' is the integer 43 in decimal notation and stands for the character '+'.

Pointer constants

- **It is dangerous to work with constant addresses.**
- **You may anyway use an integer as a constant address.**
- **But doing that lets the compiler issue you a warning message.**
- **Finally, when you run the program and try to access memory at a constant address, you are highly likely to encounter a frustrating mishap known as "Segmentation fault".**
- **It occurs when the memory is accessed at an illegal address (beyond what you are supposed to).**
- **However there is a pointer constant that is used widely. This is called NULL. A NULL pointer points to nowhere.**

Variables

- "The only constant thing is change"
- Variables help to abstract this change.
- Teacher = XYZ ;
  - here **Teacher** is a variable
  - **XYZ** is one instance of the variable, and is a constant
- A variable is an entity that has a value and is known to the program by a name,
- A variable definition associates a memory location with the variable name.
- At one time it can have only one value associated with it.

Declaring variables

- For declaring one or more variables of a given data type do the following:
- First write the data type of the variable.
- Then put a space (or any other white character).
- Then write your comma-separated list of variable names.
- At the end put a semi-colon.

Promotion and typecasting of variables

int i;

float f;

i = 5;

f = i;

The last statement assigns i to f. Since i is an integer and f is float, the conversion is automatic.

Promotion: This type of conversion, when the variable of lower type is converted to a higher type is called promotion.

Integral Promotion

int i;

char c;

c = ‘a’;

i = c;

The value present in the character variable ‘c’, i.e the ASCII code of the character ‘a’ is assigned to the integer ‘i’.

But i is typically represented using 2 bytes and c with 1 byte. Here comes the concept of sign extension.

Sign extension

- Conversion to a signed integer from character data type:
  - lower 8 bits will be the character’s value.
  - higher 8 bits will be filled with 0 or 1, depending on the Maximum Significant Bit (MSB) of the character.

  (Note: MSB is used to indicate the sign of a signed number)
- This is called sign extension.
- **Sign extension** takes place only if the variable of the higher type is *signed.*

Truncation

f=7.5

i = f

- This results in the discarding of .5. The value 7 is assigned to i.
-

Forcible Conversion

```
int i, j;
float f;
i=12; j=5;
f = i/j;
printf("%f\n",f);
```

The output is 2.0. This is because both i and j are integers, an integer division will take place.

Typecasting

- In order to have a floating division, either i or j should be float.
- We can change say i, from integer to float by typecasting, using:

  (float) i
- Thus we have to change the division line to:

  f = (float) i/j;
- **The general syntax is:**

  (type) variable_name;
  - **Typecasting can also be used to convert a higher data type to a lower type, for example:**

    (int) f