Lab Session 1

Review of C Language

Name : Durvesh Naresh Patil
PRN : 2019BTEEN00035
Sub. : RTOS

⊙ Data types

Q. What is the need of data types ?

⇒ There are different types of variables in C programming. Different types of variables require different - different memory size / space.

Data types allow a compiler to allocate the right memory space for a particular type of variable. Data types avoid unexpected behaviour of the program. It helps with code redability.

① Character :

a) What is the size of this datatype ?

⇒ 1 byte

b) What is the range of values it can hold ?

⇒ signed char : -128 to 127
Unsigned char : 0 to 255

c) What is the common purpose of this data type ?

⇒ This datatype is used to store the single character whether it is alphabet, number or any special character.

d) When to use signed char & when to use unsigned char ?

⇒ When we have to store non-negative values only, then we use unsigned char. If the variable contains 0, negative & positive values then we use the signed char.

e) If we write only char, it is taken as signed or unsigned.

⇒ It is implementation specific. But generally it is taken as signed.

On x86 systems char is generally signed. On arm system it is generally unsigned.

② Integer :

a) What is the size of this data type ?

⇒ Depends on compiler, either 2 bytes or 4 bytes. According to mingw, its size is 4 bytes.

b) What is the range of values it can hold ?

⇒ 2 Bytes :

signed = -32,768 to 32,767

unsigned = 0 to 65,535

4 Bytes:

signed = -2,147,483,648 to 2,147,483,647

unsigned = 0 to 4,294,967,295

c) What is the common purpose of this datatype ?

⇒ This data type is used to store the whole numbers & is typically used to store counts, quantities & so on.

d) When to use signed int & when to use unsigned int ?

⇒ unsigned int is used to store 0 & positive whole numbers on the other hand, we use signed int to store negative, 0 and positive whole numbers.

e) Why it is better to explicitly declare signed or unsigned ?

⇒ It is always better to explicitly declare signed or unsigned for better understanding as well as efficient use of memory

③ Float :

a) What is the size of this data type ?

⇒ ~~8 bit~~ 4 bytes

b) What is the range of values it can hold ?

⇒ 1.2 E -38 to 3.4 E +38

c) What is the common purpose of this data type ?

⇒ This data type is used to store the decimal numbers i.e. the numbers with floating point value with single precision

d) When to use signed float & when to use unsigned float ?
⇒ There is no signed float nor unsigned float.

④ long :

a) What is the size of this data type ?
⇒ 4 bytes

b) What is the range of values it can hold ?
⇒ signed : -2,147,483,648 to 2,147,483,647
unsigned : 0 to 4,294,967,295

c) What is the common purpose of this data type ?
⇒ Common purpose of long data type is range of long is greater than int. It is used to contain integer numbers that are too large to fit in the int data type.

d) Why it is better to explicitly declare signed or unsigned ?
⇒ For better understanding & for clear representation it is better to explicitly declare signed or unsigned long.

⑤ double :

a) What is the size of this data type ?
⇒ 8 byte.

b) What is the range of values it can hold ?
⇒ 2.3 E-308 to 1.7 E+308

c) What is the common purpose of this data type ?
⇒ This data type is used to store decimal numbers i.e. numbers with floating point value with double precision.

d) When to use signed double & when to use unsigned double ?
⇒ There is neither signed double nor unsigned double.

e) Why it is better to explicitly declare signed or unsigned ?
⇒ There is neither signed double nor unsigned double

⑥ void :

a) What is the meaning of void data type ? What is its need ?
⇒ void means nothing. void is used in definition ~~of~~ & prototyping of functions to indicate nothing is passed or nothing is returned

b) Can we declare variable of void type ? why ?
⇒ No, we cannot declare a variable of type void. void is a keyword which is used in function definition which returns nothing

c) What is the size of this data type ?
⇒ Does not have any size as it is a keyword.

d) What is the range of values it can hold ?
⇒ It cannot hold any value.

e) What is the common purpose of this data type ?
⇒ (i) function return type
(ii) function parameter.

⑧ Pointers in C

1) What is a pointer ?
⇒ Pointer is a variable which points / store address of another variable

2) What are the advantages of a pointer ?
⇒ (i) Allow dynamic memory allocation
(ii) Return multiple values from function
(iii) Reduce the program execution time
(iv) Reduce length & complexities of program
(v) Used in data structures like trees, linked list, etc.
(vi) Provide direct access to memory

3) How to declare a pointer ? Why do we need to specify the data type while declaring it ?
⇒ Pointer is declared with '*' sign. We need to specify the data type while declaring it because compiler should aware what the size of memory block is among others, the pointer is pointing to.

4) Write a program to illustrate use of pointer.

⇒

```
#include <stdio.h>
int main()
{
    int num = 35;
    int *ptr;
    ptr = &num;
    printf("%d", num);
    printf("\n%d", *ptr);
    printf("\n%p", &num);
    printf("\n%p", ptr);
    printf("\n%p", &ptr);
    return 0;
}
```

Output: 35
35
0x7fff5edg8c4c
0x7fff5edg8c4c
0x7fff5edg8c50

5) What is void pointer? Does that mean a pointer that points to nothing?

⇒ Void pointer is a pointer that has no associated data type with it. A void pointer can hold address of any type & can be typecasted to any type.

No it does not mean a pointer that points to nothing.

6) What is NULL pointer? Does that mean a pointer that points to nothing?

⇒ NULL pointer is the pointer that points to nothing.

7) What is use of NULL pointer ?

⇒ (i) To pass NULL pointer to a function argument when we don't want to pass any valid memory address

(ii) To initialize a pointer variable when that pointer variable is not assigned any memory address.

8) What is the need of a void pointer ?

⇒ A void pointer is really useful if the programmer is not sure about the data type of data inputted by end user. Programmer can use a void pointer to point to the location of the unknown data type.

Structure in C

1) What is a structure ?

⇒ Structure is a user-defined data type in C. It allows us to combine data of different data types together. It helps to construct a complex data type.

2) What are the advantages of a structure ?

⇒ (i) User can define their own data type which is collection of different data type

(ii) We can use array of structures or array in structures

(iii) Structure helps to improve readability of code as well as reduce redundancy.

3) How to declare a structure & why do we need to specify the data type while declaring it ?

⇒ We can declare structure using keyword 'struct'. We need to specify the data type while declaring because compiler should know about the different data types used in structure.

eg.

```
struct student
{
    char name[50];
    int roll_num;
    float percentage;
};
```

4) Write a program to illustrate use of structure.

⇒

```
#include <stdio.h>
struct Student
{
    char name[50];
    int roll_no;
    float percentage;
};
int main()
{ struct Student s1;
  s1.name = "Durvesh";
  s1.roll_no = 35;
  s1.percentage = 91.23;
  printf("%s\n%d\n%f\n",
  s1.name, s1.roll_no, s1.percentage);
  return 0;
}
```

Output : Durvesh
35
91.23

5) Read more about structure.

⇒ javatpoint.com/structure-in-c & geeksforgeeks.org/structure

C coding standards

1) What is coding standard ?

⇒ Coding standards are collections of rules & guidelines that determine the programming style, procedures & methods for a programming language

2) What are the advantages of following a coding standard ?

⇒ (i) Reduce code complexity
(ii) Reduced development cost
(iii) Easy team integration
(iv) Increase code quality, efficiency & easy for maintaining.

3) Write the common rules / guidelines for better coding.

⇒ (i) Naming conventions: How packages, classes, methods, variables, etc. should be named
(ii) File & folder naming & organization
(iii) formatting & ~~inden~~ indentation : code should be in standardized format & indentation.
(iv) commenting & documenting : This makes it easy for reviewer of your code to better understand codes methods & declarations
(v) classes & functions : This specifies how classes & functions should behave
(vi) Testing : This specifies which approach & tools should be used to test the codes.

Writing portable C code.

1) Illustrate portability:

⇒ Portability is about writing the code so that it can be easily moved (ported) to another environment so that recompiling & relinking, it will behave the same way it did originally.
(Ideally without any changes to source code itself but practically with any minimal changes)

C code 8051 μC

```
#include <REG51.H>
sbit LED = P0^0;
void delay()
{
  unsigned char i;
  for(i=0; i<100; i++);
}
void main()
{
  while(1)
  {
    LED = 0;
    delay();
    LED = 1;
    delay();
  }
}
```

C code pic18F4450 μC

```
#include <xc.h>
#define _XTAL_FREQ 20000000

void main()
{
  TRISC0 = 0;
  while(1)
  {
    RC0 = 1;
    __delay_ms(1000);
    RC0 = 0;
    __delay_ms(1000);
  }
  return;
}
```

The above C codes provide the same functionality i.e blinking LED.

2) Write the guidelines to be followed while writing portable code.

⇒ (i) Use module level API
(ii) Minimize module coupling
(iii) Encapsulate
(iv) Use ANSI-C
(v) Define a c style guide
(vi) Document the code well
(vii) Avoid bit fields.

3) Whether the RTOS code should be portable? why?

⇒ RTOS code should be portable because RTOS makes development easier for many project & it makes them more expendable, maintainable, portable & secure. Time & cost saving result.