

Recitation # 2(1/31/2002)

Topics

1. Functions
2. Arrays
3. Pointers

Functions

1.need for functions

2.function skeleton

```
type function_name(parameter type list) // function header
```

```
{
    declarations;
    statements;
}
```

3. Example

```
int add(int x,int y)
{
    return x + y;
}
```

4. Function Invocation;

```
int main(void)
{
    int a,b,c;
    a = add(b,c);
}
```

5. Function dissection

type refers to the return type of the function.
Could be int,float,char or a pointer variable.
Actual parameters, formal parameters
return statement;

6. Placing a function in a program

If you are defining a function after your main block then prototyping is required.

So what is prototyping.

consider 2 simple programs

Program 1

```
# include<stdio.h>

int add(int x,int y)
{
    return x + y;
}
int main(void)
{
    int a,b,c;
    b = 3;
    c = 5;
    a = add(b,c);
}
```

Program 2

```
# include<stdio.h>

int add (int,int);

int main(void)
{
    int a,b,c;
    a = add (b,c);
}
int add(int x,int y)
{ return x + y;}
```

```
#include<stdio.h>
int main(void)
{
    long int x = 0x0f000f00;
    char * ptr = (char *) &x;
    printf("%x \n",*ptr);
    ptr++;

    printf("%x \n",*ptr);
    ptr ++;

    printf("%x \n",*ptr);
    ptr ++;
    printf("%x \n",*ptr);
    ptr ++;

    return 0;
}
```

Output :

Solaris : SPARC, Big Endian

Linux : Intel, Little Endian

```
raghaven@caesar (~/.2021) % cc endian.c    raghaven@aurora (~/.2021) % cc endian.c
raghaven@aurora (~/.2021) % a.out          raghaven@caesar (~/.2021) % a.out
f                                           0
0                                           f
f                                           0
0                                           f
```