Functions and pointers

Discussion

- Imagine you have a locker in a bank and you want multiple people to use it what will you do?
- If you want to share information from an encyclopedia with a friend what will you do?

What is a function?

- Self-contained block of statements
- Performs coherent task
- Specific work
- Simple or complex
- Used for repetitive task
- ‘main’ is a function

Why functions?

- Avoids rewriting
- Easier to write
- Structured code
- Divide code into ‘sections’
- Easier to design and understand

- .h files contain prototypes
- Return type
- Returns control to call
- Variable arguments

Passing information to functions

- Called function arguments
- Evaluated right to left – **very important**
- variable number of arguments

Advanced function features

- Function declaration and prototype
- Calling by value
- Calling by reference
- Recursion

Function declaration and prototype

- Prototype is used to indicate the kind of function, its parameters and its return value
- Informs that such a function exists
- Declare globally once to make it accessible cross all the functions
- By default returns int

Call by value and call be reference

- Value passing creates a copy of actual arguments in the formal arguments of the function
- Pass by reference when you want function to work on the same value
- In reference passing address of actual parameters is passed on to the called function and so actual parameters can be manipulated

Pointers

- int *j : value at address contained in j is an int

Example:

int i=3;

int *j;

j=&I;

Remember:

&: address at

*: value at address

Understanding pointers

- If a is a variable and p is pointer to that variable

a-value in that variable

&a- address of variable a

*(&a) – value of address a

*p- value of address pointed by p

p- address contained in p

*(&p)-value of address p

Types of pointers

- specific
  - int
  - char
  - float
  - double
  - long
- wild
- dangling
- generic
- null

Operators on pointers

- ++p Increment pointer and return the new address
- p++ return the current address and increment pointer
- *++p value at incremented pointer
- ++*p increment value at pointer
- *p++ return the current address value and increment pointer
- *(p++) return the current address value and increment pointer
- (*p)++ return the current value and increment value at address

Applications

- Data structures
- Dynamic memory allocation
- Multiple value returning
- Function pointers
- Array manipulation
- String manipulation
- Hashing
- Sorting

Recursion

- Means doing something in terms of itself
- Possible for a function to call itself
- Eg: Factorial of number

Fact (n)= n * (n-1) * (n-2) * ….. * 1

Summary

- To avoid repetition of code and bulky programs functionally related statements are isolated into a function.
- Function declaration specifies what is the return type of the function and the types of parameters it accepts.
- Function definition defines the body of the function.
- Variables declared in a function are not available to other functions in a program. So, there won't be any clash even if we give same name to the variables declared in different functions.
- Pointers are variables which hold addresses of other variables.
- A function can be called either by value or by reference.
- Pointers can be used to make a function return more than one value simultaneously.
- Recursion is difficult to understand, but in some cases offer a better solution than loops.
- Adding too many functions and calling them frequently may slow down the program execution.

Questions

- Why functions?
- if no=5

Output:

```
int reverse( int no)
{
        if (no==0) return 0;
        else
                printf (“%d”,no);
        reverse(no--);
}
```

- Explain *ptr++,*++ptr,++*ptr, (*p)++
- Equivalent pointer expression for a[i][j][k][l]
- Given: array begins at 1002 and int size is 4

Output:

Int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12}

Value of : a[0]+1,*(a[0]+1),*(*(a+0)+1)

```
{void *vp;
char ch=74, *cp="JACK";
int j=65;
vp=&ch;
printf("%c",*(char*)vp);
vp=&j;
printf("%c",*(int*)vp);
vp=cp;
printf("%s\n",(char*)vp+2); return 0; }
```

- what error will this give

```
int a []={1,2,3,4};
int j;
for  (j=0;j<5;j++)
{
        printf("%d\n",*a);
        a++;
}
return 0;
```

Answers

- Modular, avoids rewriting
- Prints 5 till Stack overflows
- Explanation
- *(*(*(*(a+i)+j)+k)+l)
- 1006 2 2
- JACK
- Lvalue required

Contact Info

- trainers@finaldesk.com
- rishabh@finaldesk.com
- nilesh@finaldesk.com
- jignesh@finaldesk.com
- yash@finaldesk.com
- anand@finaldesk.com