Functions

Functions

- Program is collection of functions and it contains at least one function i.e main()
- **It offers code reuse(avoid rewriting).**
- **Function is key for modularized programming.**
- Large program cane be broken to small function units and it is good for maintenance.

Functions

- Function is block of statements that perform specific task.
- Function should be declared before using /calling why?
- **Functions are derived data types in Cpp to declaration is necessary before us**.
- Function name can be any valid identifier.
- Function is set of instruction and program is group of functions.

Functions

- Function is divided in to two part function header and function body.
- Syntax for declaration:

  Return Datatype Function name(arguments);

  Return type aka Function Type.

  Use of parameter/arguments names are optional.
- Default return type is int, any datatype can be used as return type.
- Before calling function it should be declared.
- Examples
  - Cal(x,y,z); int x,y,z;
  - Cal(int x,int y,int z);
  - Cal(int,int,int);

Functions

- Syntax for definition:

```
Return Datatype  Function name(formal arguments)
{
     Local variable declaration
  function code
     return  //optional
//Only one value can be returned from function
}
```

If the function is defined before main ,declaration is not needed.

- Calling function
- syntax: **fun_name(actual arguments);**
  - With value collection => var=fun(a,b,c);
  - Without collecting value => fun();

Functions

- The function which call other function is called as calling function or caller.(main)
- The function which is called from other function is called callee or called function.
- Any function can be called from any function.
- **When we call function control gets transferred to called function and after executing function code, function returns control to caller function.**

Functions

- No of actual arguments must be same as no of formal arguments.
- Names of actual and form arguments can be same as they have different memory hence changes made in formal arguments do not get reflected in actual arguments.
- Scope of formal parameters and local variables declared in function is same for function only.
- **Every time the function gets called memory gets allocated and de allocated for formal arguments and actual arguments. Static can be used for remembering value.**

Functions

- A function can return only one value to caller using return.
- More than one returns are allowed but only one returns control.
- **While passing array there is no need of major dimension of an array. Ex. Fun(int mat[][2]);**
- The function which call itself is called recursive function.
- We can call main() from other fun or recursively

Functions

- There are two types of functions
- **Standard Library functions**. Ex. getline(), get()
- The declaration of library functions is present in header files but definitions are provided by .lib file.

- **User defined functions**. Ex. add();
  - Used defined functions can be declared in header files .h and defined in .cpp file.

Story of main()

- Every c program contains at least one function called main();
- Entry and exit of every c program is main();
- C compiler declares it and programmer defines it.
- A compiler supplied routine***(called_main())*** is usually called from operating system when program execution starts.
- ***Called_main()*** routine performs two jobs
  - ***It opens stdin and stdout files.***
  - ***It passes parameter list(argc,argv,env);***

Passing values to function

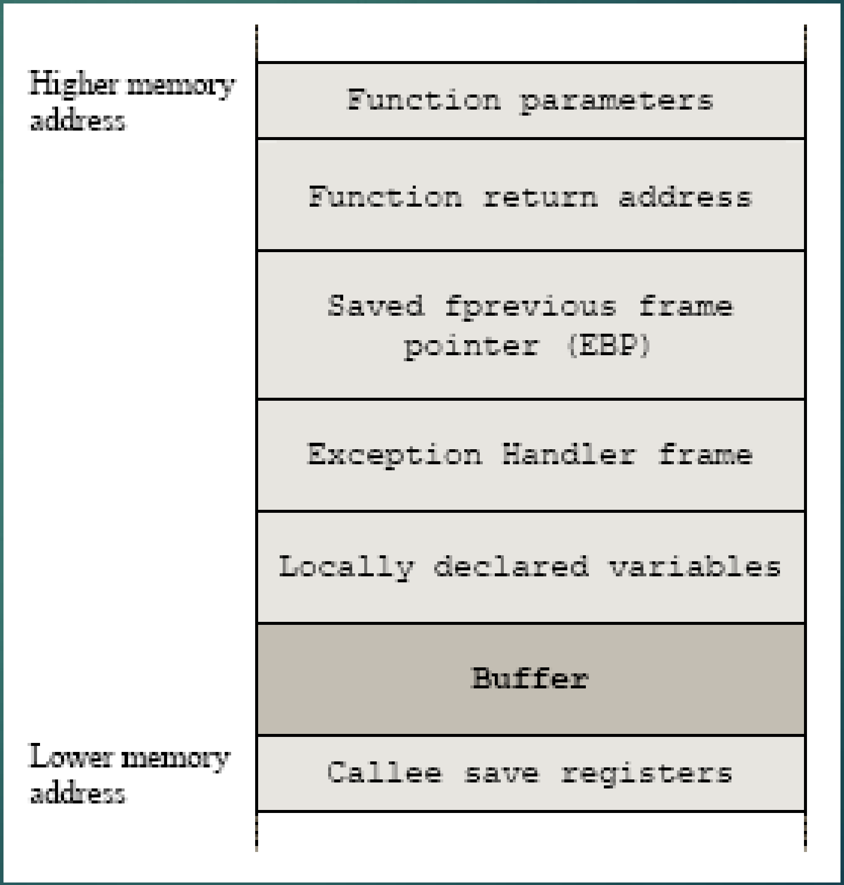
- Pass by Value/call by value
  - Actual arguments values are copied to formal argument .
  - ***Any change in formal argument does get reflected to actual arguments.***
  - Returning by value also copies value.
- Pass by reference/pass by pointer/pass by address
  - ***Address or pointers passed as arguments so changes done in formal parameters reflects in to actual arguments.***
  - While returning address be careful about local variables.

Function Calling Sequence

- Stack frame constructed during the function call for memory allocation implicitly. Explicitly, memory allocation can be requested from and released to heap area using ***malloc(), calloc(), realloc(), new, free() and delete*** respectively. A typical layout of a stack frame is shown below although it may be organized differently in different operating systems:
  - Function parameters.
  - Function's return address.
  - Frame pointer.
  - Exception Handler frame.
  - Locally declared variables.
  - Buffer.
  - Callee save registers.

Stack trace

- And the arrangement in the stack can be illustrated as shown below.



Function Calling Sequence

- As an example in Windows/Intel, typically, when the function call takes place, data elements are stored on the stack in the following way:
  - The function parameters are pushed on the stack before the function is called. The parameters are pushed from right to left.
  - ***The function return address is placed on the stack by the x86 CALL instruction, which stores the current value of the EIP register.***
  - Then, the frame pointer that is the previous value of the EBP register is placed on the stack.
  - ***If a function includes try/catch or any other exception handling construct such as SEH (Structured Exception Handling - Microsoft implementation), the compiler will include exception handling information on the stack.***
  - Next, the locally declared variables.
  - ***Then the buffers are allocated for temporary data storage.***
  - Finally, the callee save registers.

Mathematical Functions(cmath)

- Power function

  y= pow(a,b);

- Square root

  y= sqrt(a);

- Exponential

  y= exp(a);

- Logarithm

  log(x); natural log

  log10(x); log to base 10

fabs(x); absolute value
ceil(x); rounding up to nearest
floor(x); rounding down to nearest
sin, cos, tan, sinh, cosh, tanh
All Logarithm function takes and return double value

Thank You

Always write reusable code!!!!

……Functions