Passing variables to function

Variables can pass by value or by reference;

First we will discuss about passing valuable by value

In this case a copy of value of variable will pass only to function

So any change in this value will not effect on variable

void change(int a)

{

a=100;

}

int main()

{

int a=10,b=30;

chang(a);

printf("%d",a);

}

In this case value of a will not change

Because calling a function will make anew variable in the stack

Calling by reference

In this case address of variable will pass to function

So we can access his memory location and change it’s value

#include <stdio.h>

#include <stdlib.h>

void change(int &a)

{

a\*=100;

}

int main()

{

int a=10,b=30;

chang(&a);

printf("%d",a);

}

In this case a value of a will change;

In many cases we pass valuables by reference to avoid over head of memory

For example

If we had a structure that his size is 100 byte;

If we pass it by value we will make anew 100 byte in a stack

But if we pass it by reference

We will make only size of pointer which is dependent on the address bus

Of the system

Preprossesor in c;

It’s a first prossess of compilation prosses in c before compile

It has the rule to text replacement for any preprocessors directives

Like

#include <stdio.h>

#include <stdlib.h>

#define X 10

int main()

{

int z=X;

}

Preprossesor will replace X to be 10;

Other preprossesors tags

Like #if

#define NUM 10

int main(void)

{

int x = 5;

#if(NUM > 0)

x++;

printf("x = %d\n",x);

#endif

}

In this code if the NUM value >0

This code will remain

And if not >0

This code will deleted

By preprosseor

So in this case x will increase and print 6

Othor pre prossesor like macros in c

It’s look like function but its text replasment done by preprossesor

For example

#define MUX(A,B) ((A)>(B)?A:B)

int main ()

{

int a=3,b=10;

int z=MUX(a,b);

}

In this case a text replacement is done by prossesor

MUX(a,b)

Will replace by ((a)>(b)?a:b)

Some cases its better to use macros rather than function

Because no over head of function calling and context swith

But disadvantage of macros

Error tracking

Increase code size in rom

No checking of type of variable

So c++ introduce inline function rather than macros

But in c it’s not available in all compiliers

Scope and life time of the variable

Scope is the area at which we can access variable

There are three types

Block scope this variable can be used in only block that declare inside it

Like

Local variable in main()

Only use in main function

Second type is file scope

Like global variable we can use in any place in program

Third type software scope

This variable can use in many file by keyword extern

Life time

Life time is how time would variable exist in memory

There are three types

Local variable this variable will bedeallocated after end of scope

If we declare variable in for loop scope after end of for loop this variable will de allocated

And this variable exist in stack

Second type global variable is exist in all the time of execution of programme

In data segment or bss segment of ram

Like global and static variable

Third type is dynamic allocation

In this case programmer responsible for decide the life time of variable

He can free it from heap at any time

Three are some keywords that change the variable life time or scope like

Extern and static

Extern

Make a global variable can access from other files

It change scope from file scope to sw scope

Static

With global variable prevent variable to access from other files

Even if we use extern

Static with local

It make the life time of local variable is all programme execution