C Programming

Pointers

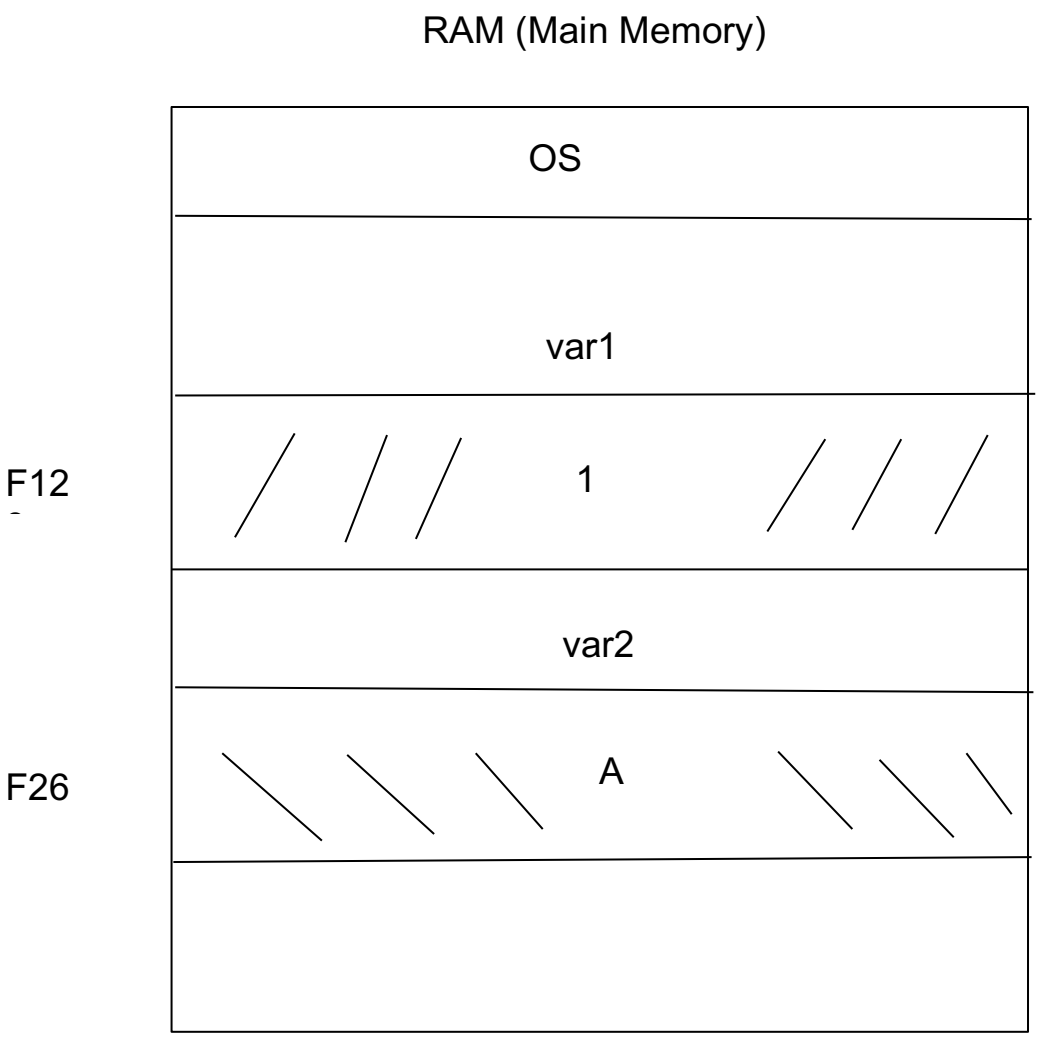
Every variable and data structure in C is stored at a unique memory address in RAM (Main Memory).

Let's look at the following code example:

```c
/*
Program to display the memory address of variables
*/

#include <stdio.h>

int main()
{
	int var1;
	char var2;


	var1 = 1;
	var2 = 'A';

	printf("\n\n\nvar1 contains %d and is stored at memory address %p\n", var1, &var1);
	printf("var2 contains %c and is stored at memory address %p\n", var2, &var2);

	return 0;

} // end main()
```

Repl 9.1: https://replit.com/@michaelTUDublin/91-Memory-address#main.c

The corresponding memory map for the above code would be as follows:



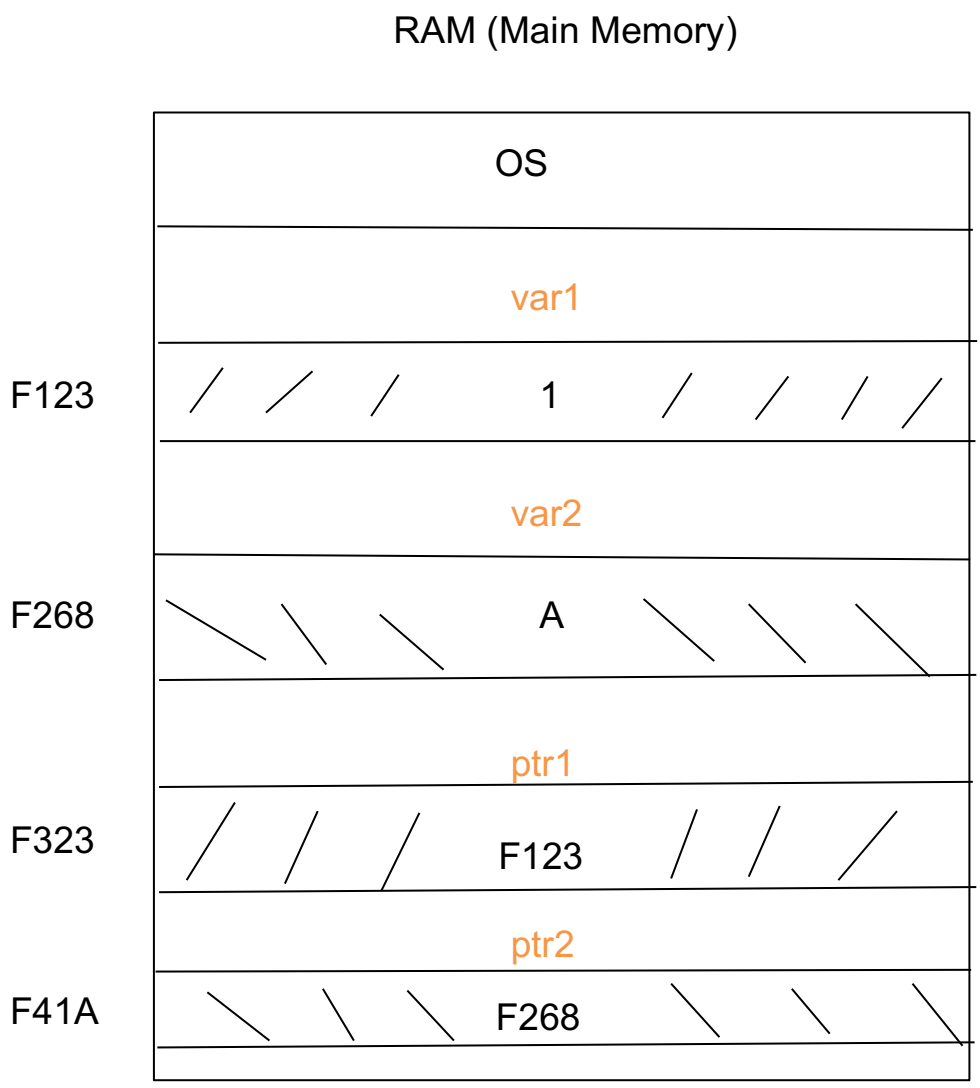
Pointer variables

A pointer variable is a variable that is used to store a memory address. Usually, a pointer variable stores the memory address of some other variable used in the same program.

```
data_type  *pointer_variable_name;
```

e.g.,

```
int *ptr;
char *my_ptr;
```



Let's modify the above code and include 2 pointer variables:

```
/*
Program to display the memory address of variables and using pointer
variables
*/

#include <stdio.h>

int main()
{
	int var1;
	char var2;
	int *ptr1;
	char *ptr2;
```

```
	var1 = 1;
	var2 = 'A';

	// Store the address of var1 inside ptr1, i.e., make ptr1 point
at var1
	ptr1 = &var1;

	// Store the address of var2 inside ptr2, i.e., make ptr2 point
at var2
	ptr2 = &var2;

	printf("\n\nvar1 contains %d and is stored at memory address
%p\n", var1, &var1);
	printf("var2 contains %c and is stored at memory address %p\n",
var2, &var2);

	printf("\nptr1 contains %p", ptr1);
	printf("\nptr2 contains %p", ptr2);

	return 0;

} // end main
```

Repl 9.2: https://replit.com/@michaelTUDublin/92-Pointer-variables#main.c

The Dereference Operator (aka the Indirection operator)

The dereference operator is used to access the contents of a regular variable whose memory address is stored in a pointer variable

Let's dereference ptr1 and ptr2 in our code above and display the result:

```
/*
Program to display the memory address of a variable, then dereference
the pointer variables and display the contents of the memory address
they point to, i.e., access the contents of the variables whose memory
address they store
```

```c
*/
#include <stdio.h>

int main()
{
	int var1;
	char var2;
	int *ptr1;
	char *ptr2;

	var1 = 1;
	var2 = 'A';

	// Make ptr1 point at var1, i.e., assign var1 memory address into ptr1
	ptr1 = &var1;
	// Make ptr2 point at var2, i.e., assign var2 memory address into ptr2
	ptr2 = &var2;

	// Display the contents and memory address of the following variables
	printf("var1 contains %d at memory address %p", var1, &var1);
	printf("\nvar2 contains %c at memory address %p", var2, &var2);

	// Display the contents of both ptr1 and ptr2
	printf("\n\nptr1 contains %p", ptr1);
	printf("\nptr2 contains %p", ptr2);

	/* Using the dereference operator to access and display the contents of var1 and var2
	*/
	printf("\n\nptr1 contains %p", ptr1);
	printf("\n*ptr1 contains %d", *ptr1);

	printf("\n\nptr2 contains %p", ptr2);
	printf("\n*ptr2 contains %c", *ptr2);

	return 0;
```

```
} // end main()
```

Repl 9.3: https://replit.com/@michaelTUDublin/93-Dereference-operator#main.c

Programming Pitfalls

1.

```
int a =10;
int b = 2;
int c;

int *ptr1;
int *ptr2;

ptr1 = &a;
ptr2 = &b;

c = a / b;

c =  *ptr1/*ptr2;   // Wrong. This is the start of a multi-line comment
c =  *ptr1 / *ptr2;   // Correct


printf("c contains %d", c);
```

2. If you declare 2 or more pointer variables on the same line, you must write the * character before EACH variable.

e.g.,

```
int *ptr1, *ptr2, *ptr3;  // Correct

int *ptr1, ptr2; // Wrong
```