18ES611 Embedded System Programming

Sarath tv

Storage Classes

- A storage class defines the ***scope*** *(visibility) and* ***life-time*** *of* ***variables*** *and/or functions* ***within*** *a C* ***Program***.
- In a C Program
  - ❖ auto
  - ❖ register
  - ❖ static
  - ❖ extern

The **auto** Storage Class

- The default storage class for all local variables.
- **Auto** can **only** be used **within functions**, i.e., **local variables**.
- **Accessed within** the **block/function.** *Scope.*
- **Garbage value** by **default**.

{

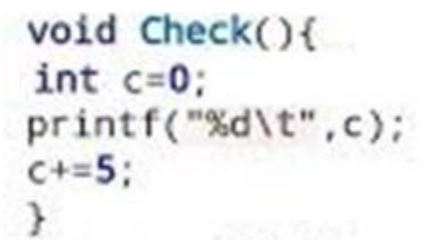
}

The register Storage Class

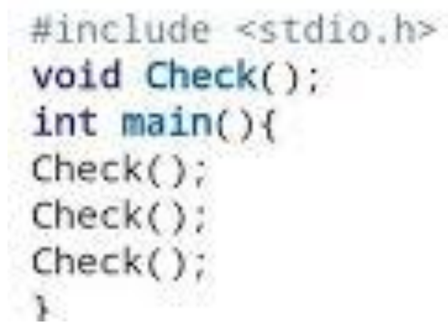
- To define local variables - stored in a register instead of RAM.
- **Maximum size equal to the register size** (usually one word)
- Can't have the unary '&' operator applied to it.
- **Variables** that **require quick access**.
- **"It should also be noted that defining 'register' does not mean that the variable will be stored in a register. "**
- Faster accessibility than a normal variable.
- But only a few variables can be placed inside registers.
- In using loops.

The static Storage Class

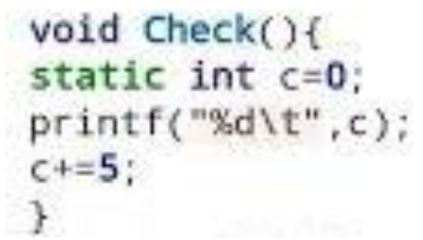
- Keep a local variable in existence during the life-time of the program.
- To maintain their values between function calls.
- May also be applied to global variables. Restricts to the file in which it is declared.
- If declared inside a function, it remains into existence till the end of the program and not get destroyed as the function exists (as in auto).
- If declared outside all the functions in a program, it can be used only in the program in which it is declared and is not visible to other program files.



```
void Check(){
 int c=0;
printf("%d\t",c);
c+=5;
}
```



```
#include <stdio.h>
void Check();
int main(){
Check();
Check();
Check();
}
```



```
void Check(){
static int c=0;
printf("%d\t",c);
c+=5;
}
```

The extern Storage Class

- To give a reference of a global variable that is visible to ALL the program files.
- When you use **'extern'**, the **variable cannot** be **initialized.**
- **Multiple files,** define a **global variable** or function, to be used in **other files**, then extern is used to give reference of defined variable or function.

•Just for understanding, extern is used to declare a global variable or function in another file.

- To **inform** the **compiler** that this **variable** is **declared somewhere else**.
- **Does not allocate storage** for variables.

Main.c

```c
#include <stdio.h>

extern int varOne ;

int main()

{

printf("value of the global variable is =
%d\n", varOne);


return 0;


}
```

Someotherfile.c

```c
int varOne=49;
```

Which storage class should be used and when

- To improve the speed of execution of the program and to carefully use the memory space occupied by the variables, following points should be kept in mind while using storage classes:
- We should use **static storage** class only when we want the value of the **variable** to **remain same** every time we call it using **different function calls**.
- We should use **register** storage class only **for** those **variables** that are used in our **program** very **often**. CPU **registers** are **limited** and thus should be used carefully.
- We should use external or **global** storage class only for those variables that are being used by **almost all** the **functions** in the program.
- If we do not have the purpose of **any** of the **above mentioned** storage classes, then we should use the **automatic storage class**.

THANK YOU!!!!!