Programming and Data Structures - I

Lecture 4

Kripabandhu Ghosh

CDS, IISER Kolkata

CONSTANT, VARIABLES AND DATA TYPES

Storage Classes

Name	Meaning
auto (default)	Local variable known only to the function in which it is declared
static	Local variable which exists and retains its value even after control is returned to the calling function
extern	Global variable known to all functions
register	Local variable stored in a register

- Global (extern) and static variables are automatically initialized to zero
- Local variables have garbage values unless explicitly initialized

Broad variable types

- **Local** variables are defined within a function or block and their scope (region where they are available for use) is limited to the same
- **Global** variables are defined outside all the functions/block
  - Scope from the point of declaration till the end of the program
  - Available to all the functions in this scope

Local vs Global variable

If a local and a global variable have the *same name*, local variable will have preference over the latter.

auto variable

- Local to a function or block
- Created when the function is called and destroyed automatically when the function ends
- The keyword **auto** may or may not be used for declaration

auto: function

Program

```
#include<stdio.h>

void function1();
void function2();

void main()
{
   int var = 5;
   printf("Value in main : %d\n", var);
   function1();
   function2();
}
void function1()
{
   int var = 1;
   printf("Value in function1 : %d\n", var);
}
void function2()
{
   int var = 2;
   printf("Value in function2 : %d\n", var);
}
```

auto: function

Program

```
#include<stdio.h>

void function1();
void function2();

void main()
{
  int var = 5;
  printf("Value in main : %d\n", var);
  function1();
  function2();
}
void function1()
{
  int var = 1;
  printf("Value in function1 : %d\n", var);
}
void function2()
{
  int var = 2;
  printf("Value in function2 : %d\n", var);
}
```

Output

```
Value in main : 5
Value in function1 : 1
Value in function2 : 2
```

auto: function (explicit declaration)

Program

```
#include<stdio.h>

void function1();
void function2();

void main()
{
   auto int var = 5;
   printf("Value in main : %d\n", var);
   function1();
   function2();
}
void function1()
{
   auto int var = 1;
   printf("Value in function1 : %d\n", var);
}
void function2()
{
   auto int var = 2;
   printf("Value in function2 : %d\n", var);
}
```

auto: function (explicit declaration)

Program

```
#include<stdio.h>

void function1();
void function2();

void main()
{
   auto int var = 5;
   printf("Value in main : %d\n", var);
   function1();
   function2();
}
void function1()
{
   auto int var = 1;
   printf("Value in function1 : %d\n", var);
}
void function2()
{
   auto int var = 2;
   printf("Value in function2 : %d\n", var);
}
```

Output

```
Value in main : 5
Value in function1 : 1
Value in function2 : 2
```

auto: block

Program

```
#include<stdio.h>

void main()
{
   int var = 5;
   printf("Value in main : %d\n", var);
   {//block 1
      printf("Value in block 1 : %d\n", var);
      {//block 2
         printf("Value in block 2 : %d\n", var);
         {//block 3
            printf("Value in block 3 : %d\n", var);
         }
      }
   }
}
```

auto: block

Program

```
#include<stdio.h>

void main()
{
   int var = 5;
   printf("Value in main : %d\n", var);
   {//block 1
      printf("Value in block 1 : %d\n", var);
      {//block 2
         printf("Value in block 2 : %d\n", var);
         {//block 3
            printf("Value in block 3 : %d\n", var);
         }
      }
   }
}
```

Output

```
Value in main : 5
Value in block 1 : 5
Value in block 2 : 5
Value in block 3 : 5
```

auto: block (contd.)

Program

```
#include<stdio.h>

void main()
{
  int var = 5;
  printf("Value in main : %d\n", var);
  {//block 1
    int var = 6;
    printf("Value in block 1 : %d\n", var);
    {//block 2
      int var = 7;
      printf("Value in block 2 : %d\n", var);
      {//block 3
        int var = 8;
        printf("Value in block 3 : %d\n", var);
      }
      printf("Value in block 2 again : %d\n", var);

    }
    printf("Value in block 1 again : %d\n", var);
  }
  printf("Value in main : %d\n", var);
}
```

auto: block (contd.)

Program

```
#include<stdio.h>

void main()
{
  int var = 5;
  printf("Value in main : %d\n", var);
  {//block 1
    int var = 6;
    printf("Value in block 1 : %d\n", var);
    {//block 2
      int var = 7;
      printf("Value in block 2 : %d\n", var);
      {//block 3
        int var = 8;
        printf("Value in block 3 : %d\n", var);
      }
      printf("Value in block 2 again : %d\n", var);

    }
    printf("Value in block 1 again : %d\n", var);
  }
  printf("Value in main : %d\n", var);
}
```

Output

```
Value in main : 5
Value in block 1 : 6
Value in block 2 : 7
Value in block 3 : 8
Value in block 2 again : 7
Value in block 1 again : 6
Value in main again : 5
```

Global variable (with a local variable)

Program

```
#include<stdio.h>

void function1();
void function2();

int var = 5;//Global variable

void main()
{
   printf("Value in main : %d\n", var);
   function1();
   function2();
}

void function1()
{
   int var = 1; //Local declaration
   printf("Value in function1 : %d\n", var);
}

void function2()
{
   printf("Value in function2 : %d\n", var);
}
```

Global variable (with a local variable)

Program

```
#include<stdio.h>

void function1();
void function2();

int var = 5;//Global variable

void main()
{
   printf("Value in main : %d\n", var);
   function1();
   function2();
}

void function1()
{
   int var = 1; //Local declaration
   printf("Value in function1 : %d\n", var);
}

void function2()
{
   printf("Value in function2 : %d\n", var);
}
```

Output

```
Value in main : 5
Value in function1 : 1
Value in function2 : 5
```

External declaration

Program

```
#include<stdio.h>
void function1();
void function2();
void main()
{
   printf("Value in main : %d\n", var);
   function1();
}
int var = 5;//Global variable
void function1()
{
   printf("Value in function1 : %d\n", var);
}
```

External declaration

Program

```
#include<stdio.h>
void function1();
void function2();
void main()
{
   printf("Value in main : %d\n", var);
   function1();
}
int var = 5;//Global variable
void function1()
{
   printf("Value in function1 : %d\n", var);
}
```

Output

error: 'var' undeclared (first use in this function)

External declaration (resolved)

Program

```
#include<stdio.h>

void function1();
void function2();

void main()
{
   extern int var;
   printf("Value in main : %d\n", var);
   function1();
}

int var = 5;//Global variable

void function1()
{
   printf("Value in function1 : %d\n", var);
}
```

External declaration (resolved)

Program

```
#include<stdio.h>

void function1();
void function2();

void main()
{
   extern int var;
   printf("Value in main : %d\n", var);
   function1();
}

int var = 5;//Global variable

void function1()
{
   printf("Value in function1 : %d\n", var);
}
```

Output

```
Value in main : 5
Value in function1 : 5
```

External declaration (resolved)

Program

```
#include<stdio.h>

void function1();
void function2();

void main()
{
   extern int var;
   printf("Value in main : %d\n", var);
   function1();
}

int var = 5;//Global variable

void function1()
{
   printf("Value in function1 : %d\n", var);
}
```

Output

```
Value in main : 5
Value in function1 : 5
```

Also available across files

Static variable

Program

```
#include<stdio.h>

void function1();
void function2();

void main()
{
   function1();
   function1();
   function1();
   function2();
   function2();
   function2();
}

void function1() //Auto
{
   int var = 0;
   var++;
   printf("Value in function1 : %d\n", var);
}

void function2() //Static
{
   static int var = 0;
   var++;
   printf("Value in function2 : %d\n", var);
}
```

Static variable (contd.)

Output

```
Value in function1 : 1
Value in function1 : 1
Value in function1 : 1
Value in function2 : 1
Value in function2 : 2
Value in function2 : 3
```

Static variable : properties

Output

- **Local**: Retain values between function calls
- **Global**: Same as a normal global variable except, unlike *extern*, not available across files

register variable

Declaration

register int var;

- Keeps value in a register than a normal memory location
- supposed to be faster access

OPERATORS

Operators

Type	Examples
Arithmetic	+ (addition/unary plus) - (subtraction/unary minus) * (multiplication) / (division) % (modulo division)
Relational	<, <=, >, >=, ==, ! =
Logical	&& (AND), | (OR)
Assignment	=
Increment and decrement	++,
Conditional	?:

Thanks