

Fundamentals of C Language

By Sharanam Chotai



Programming Language

- Syntax:
 - Set of rules
- Semantics:
 - Meaning of the syntax
- Compilation/Interpretation:
 - Translation to Machine Language
- Automation:
 - Repetitive and complex tasks.
- Problem-Solving:
 - Focus on solution, not on tools.



C language

- Closer to real hardware
- To create OS (Unix in 1969)
by Dennis Ritchie
at the Bell Laboratories in 1972
- Procedural programming language
(general-purpose)

Programming Paradigm

- Way of thinking about and structuring code

“Most programming languages support multiple paradigms. A programmer's growth involves learning to choose the appropriate programming language and paradigm for the situation.”

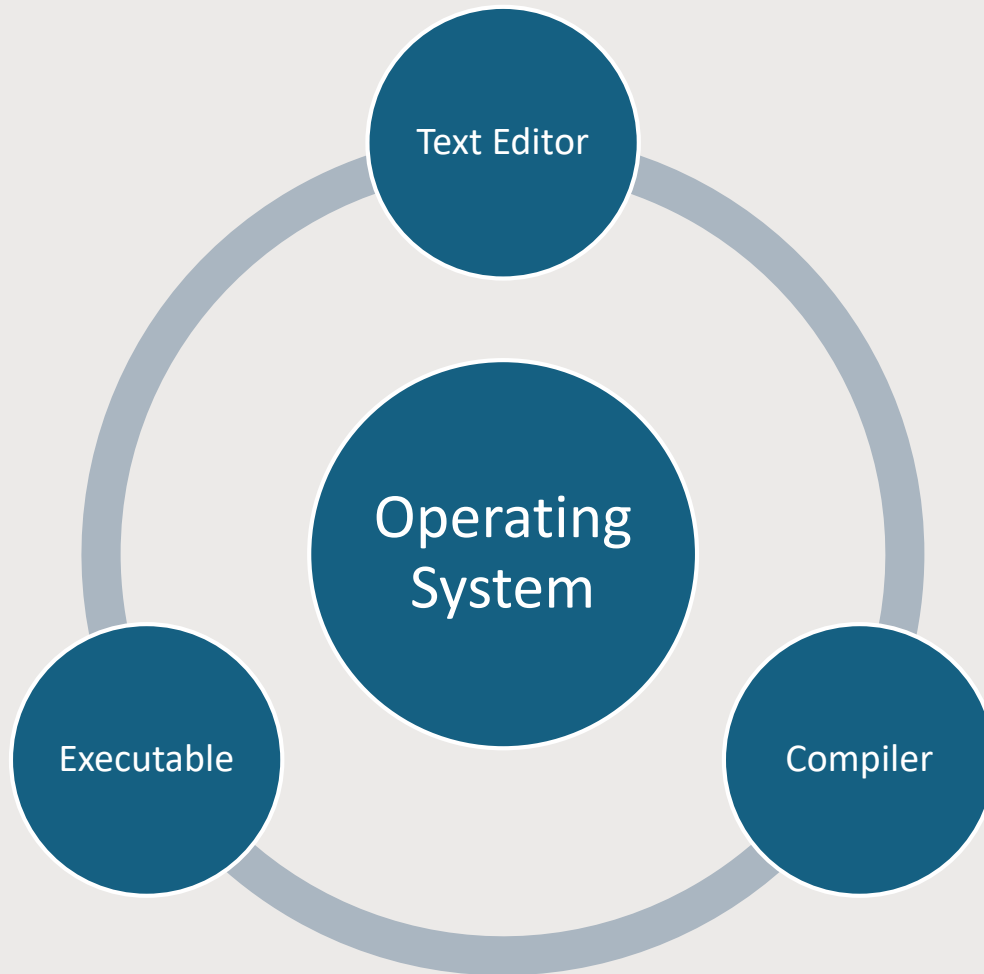
- Mooc.fi

C as a procedural language:

- Function-based
- Sequential execution
- Structured programming



Required Tools



bo C++ IDE

File Edit Search Run Compile Debug Project Options Window Help

\DOS\TURBOC\TCPP30\EXAMPLES\CIRCLE.CPP 1

```
setcolor<TempColor>; // set color back to current color

void Circle::Expand(int ExpandBy)
{
    Hide();
    Radius += ExpandBy;
    if (Radius < 0)
        Radius = 0;
    Show();
};

void Circle::Contract(int ContractBy)
{
    Expand(-ContractBy);
};

void Circle::MoveTo(int X, int Y)
{
    Hide();
    X = NewX;
    Y = NewY;
    Show();
};

int main()
{
    // initialize the graphics system
    auto int graphdriver = DETECT, graphmode;
    initgraph(&graphdriver, &graphmode, "..\\..\\bgi");

    Circle MyCircle(100, 200, 50); // declare a circle object
    MyCircle.Show(); // show it
    getch(); // wait for keypress
    MyCircle.MoveTo(200, 250); // move the circle (tests hide and show also)

    getch();
    MyCircle.Expand(50); // make it bigger
    getch();
    MyCircle.Contract(75); // make it smaller
    getch();
    closegraph();
    return 0;
}
```

Help 3=11

auto <keyword>

Defines a local variable as having a local lifetime.

Syntax: [auto] <data definition>;

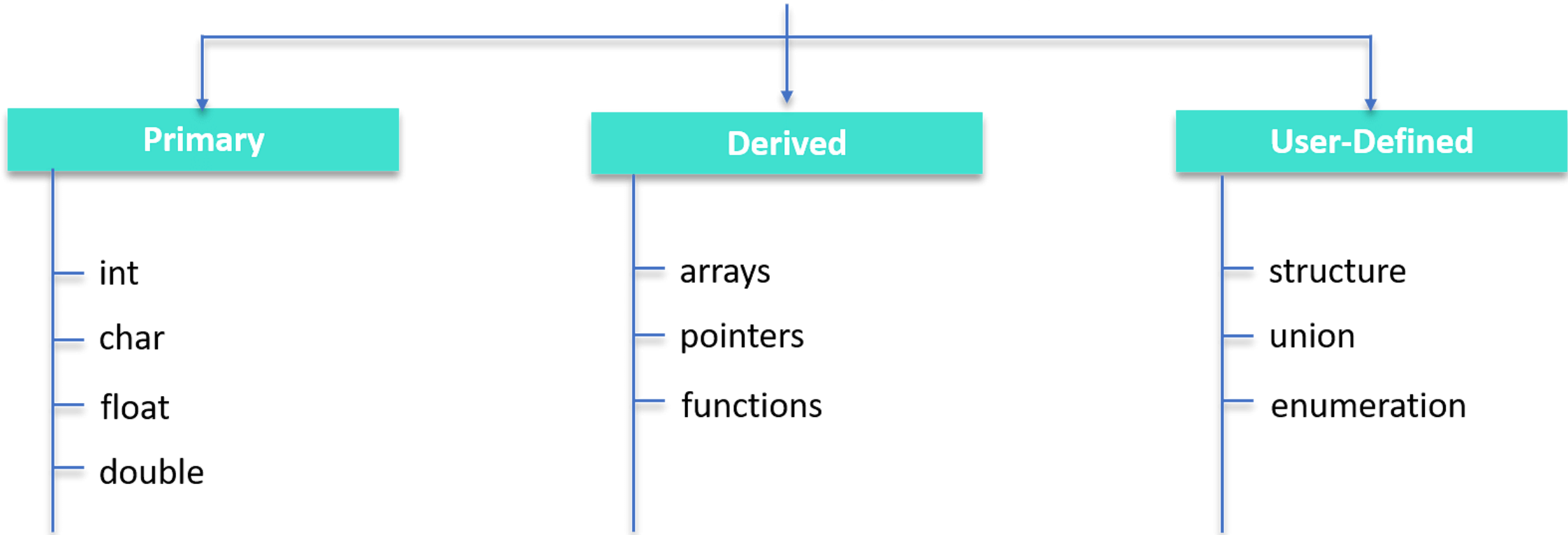
This is the default for local variables and therefore is rarely used.

Example:

```
int main(int argc, char **argv)
{
    auto int i;

    i = 5;
    return i;
}
```

DATA TYPES IN C



Types of Operators

1. Arithmetic Operators: +, -, *, /, %, ++, --
2. Relational Operators: ==, !=, >, <, >=, <=
3. Logical Operators: &&, ||, !
4. Bitwise Operators: &, |, ^, ~, <<, >>
5. Assignment Operators: =, +=, -=, *=, /=, %=, <<=, >>=, &=, |=, ^=
6. Conditional Operator: ?:
7. Cast Operator: (type)
8. Comma Operator: ,
9. Address Operator: &
10. Dereference Operator: * (also known as Indirection Operator or Value at Address Operator)

Decision Making Statements

1. if statement
2. if-else statement
3. nested if-else statement
4. else-if ladder
5. switch statement



Looping statements

1. for
2. while
3. do...while

Branching statements

1. break
2. continue

Function in C

- Block of code which only runs when it is called
- Aspects of using function
 - Function Declaration
 - Function Definition
 - Function Calls
- Type of arguments:
 - Pass by Value
 - Pass by Reference

The diagram shows the function declaration `int sum (int a , int b);` with several annotations. Above the code, there are two labels: 'return type' with an arrow pointing to 'int', and 'Parameter Type' with an arrow pointing to 'int'. Below the code, there are three labels: 'Function Name' with an arrow pointing to 'sum', 'Parameter Name' with an arrow pointing to 'a', and 'Ending Statement Semicolon' with an arrow pointing to ';'. The entire code is written in green.

```
return type      Parameter Type
  ↑              ↑
int sum ( int a , int b );
  ↓             ↓             ↓
Function Name  Parameter Name Ending Statement Semicolon
```

Quiz time

1. What is the return type of scanf and printf functions?
2. What is the difference between = and == in C?
3. `int a = 5, b = 10; printf("%d", a++ + ++b);`
4. Which loop is also called exit-controlled loop?