Tokens, Expressions and Control Structure

Prepared by Deepali Sonawane, MIT-WPU, Pune

Tokens

- **C++ tokens** are the smallest individual units of a program, that the compiler recognizes and processes. Together, they form the syntax of C++ code, the same as combining words to form a sentence.
- C++ is the **superset of C** and so most constructs of C are legal in C++ with their meaning and usage unchanged. So tokens, expressions, and data types are similar to that of C.
- Here are the following C++ tokens given below
  - Keywords
  - Identifiers
  - Constants
  - Variables
  - Operators

Keywords

- **Keywords** are reserved words which have **fixed meaning**, and its meaning cannot be changed.
- The **meaning** and **working** of these keywords are already known to the **compiler**.
- C++ has **more numbers** of keyword than C, and those extra ones have special working capabilities.
- List of some commonly used keywords
  - **Control flow:** if, else, switch, case, break, continue, return, goto
  - **Data types:** int, char, double, float, bool, void, long, short
  - **Storage classes:** static, extern, mutable, register, volatile
  - **Object-oriented programming:** class, struct, public, private, protected, virtual, this, friend
  - **Exception Handling** : try, catch, throw, thrown
  - **Other:** namespace, using, typedef, const, sizeof, typeid, template, new, delete

Identifiers

- **Identifiers** are names given to different entries such as variables, structures, and functions, classes, objects, arrays, etc.
- Also, identifier names should have to be **unique** because these entities are used in the execution of the program.

Identifier naming conventions

- Only alphabetic characters with a **letters** (A-Z, a-z), **digits** (0-9), and **underscores** (_) are permitted.
- The first letter must be an **alphabet** or **underscore (_),** not a number.
- Identifiers are **case sensitive**.
- **Reserved keywords** can not be used as an identifier's name.

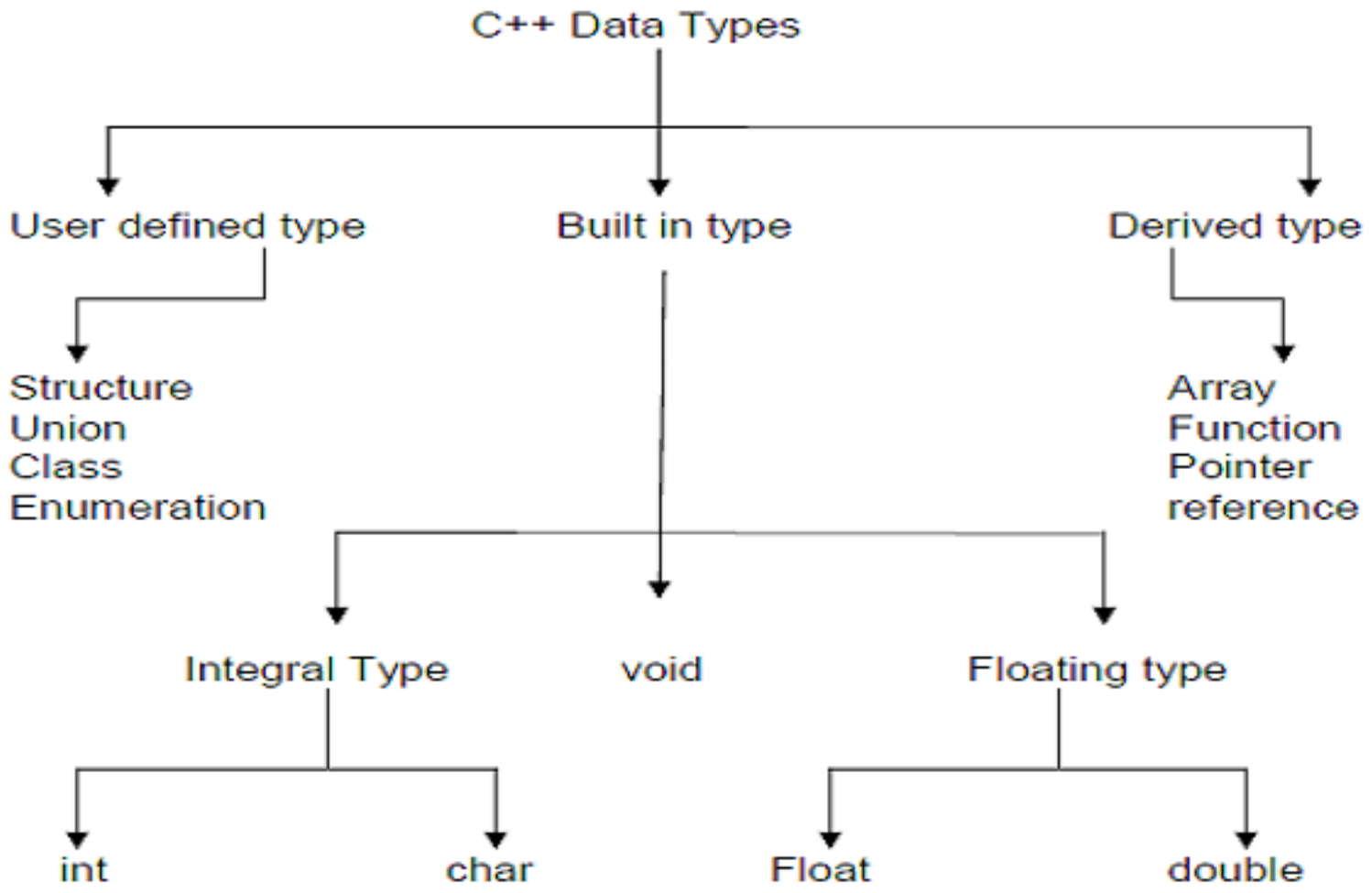
Constants

- **Constants** are like a variable, except that their **value never changes** during execution once defined.
- Constants contains **fixed value**.
- There are two other different ways to define constants in C++.
  - By using **const** keyword

  Syntax: const [data_type] [constant_name]=[value];
  - By using #define preprocessor

  Syntax: #define contant_name value

Data Types


C++ Data Types
User defined type
Built in type
Derived type
Structure
Union
Class
Enumeration
Array
Function
Pointer
reference
Integral Type
void
Floating type
int
char
Float
double

Data Types and Bytes required

- void : 0 Byte
- char : 1 Byte
- short : 2 Bytes
- long : 4 Bytes
- int : 2 Bytes [-32768 to 32767]
- float : 4 Bytes [7 precision]
- double : 8 Bytes [14 precision]
- long double : 10 Bytes
- bool : 1 Byte

Variables

- A variable is a **meaningful name** of data storage location in computer memory.
- When using a variable you refer to **memory address** of computer.
- **Syntax**: data_type variablename [= value];
- Example:

```
main()
{
        int num = 23;
        cout<<"Value of num ="<<num;
}
```

Reference variable

- A reference is a variable name that is a **duplicate of an existing variable**.
- It provides a technique of creating more than one name to designate the same variable.
- **Syntax**: data_type &ReferenceName= Variablename;
- Example:

```
main()
{
        int num = 23;
        int &Rnum = num;
        cout<<"Value of num ="<<num;
        cout<<"Value of Rnum="<<Rnum;
}
```

Operators of CPP

1. **Arithmetic Operators**: Arithmetic operators are used to perform arithmetic operations. +, -, *, /, % are arithmetic operators.
2. **Relational Operators:** Relational operators are used for comparison. <, <=, >, >=, ==, != are relational operators.
3. **Logical Operators:** Logical operators are used for combining two conditions. &&, ||, ! are logical operators.
4. **Increment and Decrement Operators:** Increment operator (++) increase the value of variable by 1 and decrement operator (--) decrease the value of variable by 1.
5. **Assignment Operator:** Assignment operator (=) assign value to variable. +=, -=, *=, /=, %= are short-hand assignment operators.
6. **Conditional Operators:** ? : is a conditional operator. It is ternary operator also.

   Synatx: (condition) ? true part : false part;

New operators of CPP

Operator	Meaning
endl	Line feed operator
new	Memory allocation operator
delete	Memory deallocation operator
>>	Extraction Operator
<<	Insertion Operator
::	Scope resolution Operator
::*	Pointer to member operator
->*	Pointer to member operator

Control Structure of CPP

- **Decision/ Selection Making Statements**
  1. If statement
  2. If-else statement
  3. Nested if statement
  4. Else – if ladder statement
  5. Switch Statement
- **Loop / Iterative Statements**
  1. While loop
  2. Do-while loop
  3. For loop
- **Jump Statements**
  1. Break statement
  2. Continue statement
  3. goto statement

Decision Making Statement

1) if statement

Syntax : if (condition)

```
{
        statements;
}
```

2) if-else statement

Syntax : if(condition)

```
{
        statement1;
}
else
{
        statement2;
}
```

Decision Making Statement

3) Nested if statement

Syntax:

```
if(condition1)
        {
                if(condition2)
                { statement1; }
                else
                { statement2; }
        }
        else
        {
                statement3;
        }
```

Decision Making Statement

4) else if ladder statement

Syntax: if(condition1)

```
        statement1;
    else if(condition2)
        statement2;
    else if(condition3)
        statement3;
        .........
    else
        statement n;
```

Decision Making Statement

5) switch statement

Syntax: switch(expression)

```
        {
            case label 1 : statements;
                           break;
            case label 2 : statements;
                           break;
                .....
            default : statements;
                           break;
        }
```

Loop Statements /Iterative statements

1) while loop

Syntax:

```
initialization;
while(condition)
{
        statements;
        update statement;
}
```

Loop Statement /Iterative statements

2) do while loop

Syntax:

```
initialization;
do
{
        statements;
        update statement;
} while(condition);
```

Loop Statement / Iterative statements

3) for loop

Syntax: for(initialization ; condition ; update statement)

```
{
        statements;
}
```

Example : To display numbers from 1 to 10.

```
for(i=1 ; i<=10 ; i++)
{
        cout<<i<<"\n";
}
```

Jump Statement

1) break statement : break statement is used to jump out from the loop.

Syntax: break;

Example:

```
for(i=1;i<=10;i++)
{
        if(i%2==0)
                break;
        cout<<i;
}
```

Jump Statement

2) continue statement : The continue statement breaks one iteration in the loop, if a specified condition occurs, and continues with the next iteration in the loop.

Syntax: continue;

Example:

```
for(i=1;i<=10;i++)
{
        if(i%2==0)
                continue;
        cout<<i;
}
```

Jump Statement

3) goto statement

- A **goto** statement provides an unconditional jump from the goto to a labeled statement in the same function.
- The **goto** statement in C++ is a kind of control flow construct that lets the program jump directly to a given statement within the same function.
- The **goto** statement transfers control to a given label in the function. A label is defined with an identifier followed by a colon **(:)**.

Syntax:

goto label;

. . .

. . .

label: statement;

Thank you