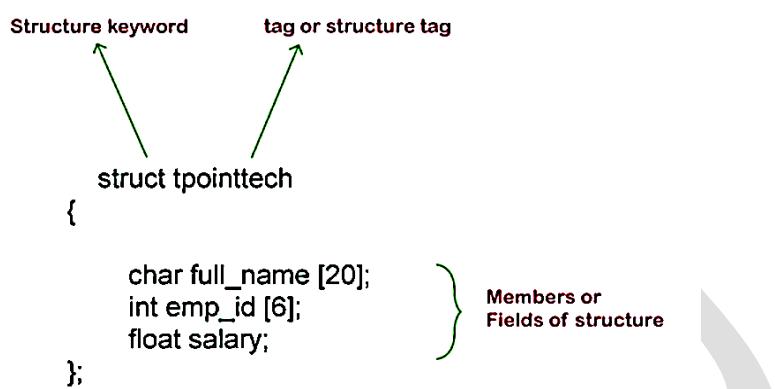


UNIT – 3

Structures (स्ट्रक्चर)

Structure एक **User-Defined Data Type** है, जो विभिन्न प्रकार के डेटा को एक साथ संगठित करने की अनुमति देता है।

यह विशेष रूप से तब उपयोगी होता है जब हमें किसी इकाई (जैसे Student, Employee, Product आदि) के विभिन्न गुणों को एक साथ संग्रहित करना होता है।



Structure का Syntax (Structure Syntax)

```

struct StructureName {
    dataType member1;
    dataType member2;
    ...
    dataType memberN;
};
  
```

- **struct:** कीवर्ड जो Structure को परिभाषित करता है।
- **StructureName:** Structure का नाम (उपयोगकर्ता द्वारा चुना गया)।
- **member1, member2,..., memberN:** Structure के सदस्य (Members)।
- **dataType:** सदस्य का डेटा प्रकार (int, float, char आदि)।

उदाहरण:

```

struct Student {
    int rollNo;
    char grade;
    float marks;
};
  
```

Structure का नाम: **Student**

Members: **rollNo** (int), **grade** (char),
marks (float)

Structure Variable की घोषणा (Declaring Structure Variable)

Structure को परिभाषित करने के बाद, हम इसके variables को निम्नलिखित तरीके से घोषित कर सकते हैं:

Student s1;

◆ Structure Members को Access करना (Accessing Structure Members)

Structure के members को access करने के लिए **Dot (.) Operator** का उपयोग किया जाता है।

उदाहरण:

```
s1.rollNo = 101;  
s1.grade = 'A';  
s1.marks = 89.5;
```

```
#include <iostream>  
using namespace std;  
  
struct Student {  
    int rollNo;  
    char grade;  
    float marks;  
};  
int main() {  
    Student s1;  
    s1.rollNo = 101;  
    s1.grade = 'A';  
    s1.marks = 89.5;  
  
    cout << "Roll No: " << s1.rollNo << endl;  
    cout << "Grade: " << s1.grade << endl;  
    cout << "Marks: " << s1.marks << endl;  
    return 0;  
}
```

C++ Structures में Member Functions

C++ में, Structure के अंदर भी functions हो सकते हैं, जो Structure के data elements पर सीधे operations को सक्षम बनाते हैं।

ये Member Functions सामान्य functions की तरह ही होते हैं, लेकिन इन्हें Structure के scope में परिभाषित किया जाता है।

Structures में Member Functions के लाभ

Structure में data और behavior को एक साथ रखने से यह अधिक अनुकूल (adaptable) बन जाता है, बिलकुल classes की तरह।

C++ Structure में कई class components होते हैं, जैसे:

- **Access Specifier** (जैसे private, public)
- **Constructor** (object बनाने के लिए)
- **Destructor** (object को नष्ट करने के लिए)
- और कई अन्य सुविधाएँ।

```

#include <iostream>
using namespace std;

struct Student {
    int rollNo;
    char grade;
    float marks;

    // Member Function to display data
    void display() {
        cout << "Roll No: " << rollNo << endl;
        cout << "Grade: " << grade << endl;
        cout << "Marks: " << marks << endl;
    }

    // Constructor to initialize values
    Student(int r, char g, float m) {
        rollNo = r;
        grade = g;
        marks = m;
    }

    // Destructor
    ~Student() {
        cout << "Destructor called for Roll No: " << rollNo << endl;
    }
};

int main() {
    // Creating an object using constructor
    Student s1(101, 'A', 89.5);

    // Calling member function
    s1.display();

    return 0;
}

```

C++ में Array of Structures

C++ में हम कई structure variables को एक साथ क्रमबद्ध तरीके से **Array of Structures** में संग्रहीत कर सकते हैं।

यह विशेष रूप से तब उपयोगी होता है जब हमें एक ही प्रकार के कई रिकॉर्ड को संगठित रूप से संग्रहित करना हो।

Structure Array में डेटा तक पहुँच (Accessing Data)

Dot (.) Operator और **Array Index** का उपयोग करके individual members तक पहुँचा जा सकता है।

Array में प्रत्येक तत्व (element) Structure का एक instance दर्शाता है।

उदाहरण:

```
#include <iostream>
using namespace std;

struct Student {
    int rollNo;
    char grade;
    float marks;
};

int main() {
    Student s[3]; // Structure का Array (3 Students के लिए)

    // Read Structure Array
    for (int i = 0; i < 3; i++) {
        cout << "Enter Roll No, Grade और Marks for Student " << i + 1 << ": ";
        cin >> s[i].rollNo >> s[i].grade >> s[i].marks;
    }

    // Show Structure Array
    cout << "\nStudents Details:\n";
    for (int i = 0; i < 3; i++) {
        cout << "Student " << i + 1 << " -> Roll No: " << s[i].rollNo
            << ", Grade: " << s[i].grade << ", Marks: " << s[i].marks << endl;
    }

    return 0;
}
```

आउटपुट:/OUTPUT :

```
Enter Roll No, Grade और Marks for Student 1: 101 A 85.5
Enter Roll No, Grade और Marks for Student 2: 102 B 75.0
Enter Roll No, Grade और Marks for Student 3: 103 A 92.3
```

```
Students Details:
```

```
Student 1 -> Roll No: 101, Grade: A, Marks: 85.5
Student 2 -> Roll No: 102, Grade: B, Marks: 75.0
Student 3 -> Roll No: 103, Grade: A, Marks: 92.3
```

Array of Structures का उपयोग (Applications):

1. **Student Records:** एक कक्षा के सभी छात्रों का डेटा संग्रहित करना।
2. **Employee Database:** एक कंपनी के कर्मचारियों की जानकारी संग्रहीत करना।
3. **Product Details:** इन्वेंट्री में उत्पादों की सूची रखना।

Union in C++

Union एक user-defined data type है, जो एक से ज्यादा variables को एक ही memory location में store करता है।

इसका मतलब: सभी members एक ही memory share करते हैं, लेकिन एक समय में सिर्फ एक ही member वैध (valid) होता है।

Syntax (सिंटैक्स)

```
union Data {
```

```
    int i;  
    float f;  
    char ch;  
};
```

यहाँ Data नाम का union तीन members रखता है:

- i (integer)
- f (float)
- ch (character)

Example (उदाहरण)

```
#include <iostream>  
using namespace std;  
  
union Data {  
    int i;  
    float f;  
    char ch;  
};  
  
int main() {  
    Data d;  
    d.i = 10;  
    cout << "i: " << d.i << endl;  
  
    d.f = 3.14;  
    cout << "f: " << d.f << endl;  
  
    d.ch = 'A';  
    cout << "ch: " << d.ch << endl;  
  
    // अब फिर से i print करेंगे  
    cout << "i (again): " << d.i << endl;  
  
    return 0;  
}
```

Output:

```
i: 10  
f: 3.14  
ch: A  
i (again): (garbage value)
```

क्योंकि f और ch ने i की memory overwrite कर दी।

Union और Structure में अंतर (Difference between Union and Struct)

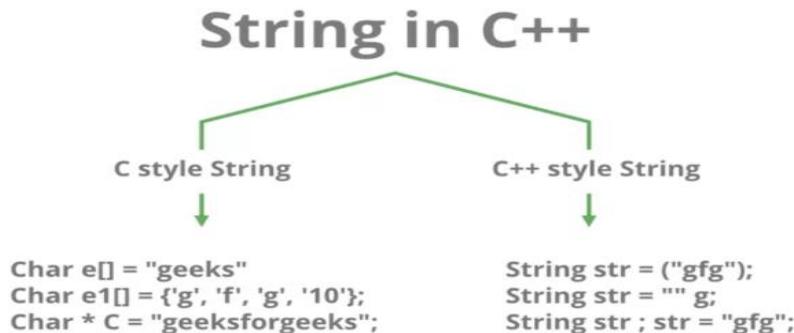
| Feature | Structure (struct) | Union (union) |
|---------------|-----------------------------------|---|
| Memory | सभी members के लिए अलग-अलग memory | सभी members एक ही memory share करते हैं |
| Size | सभी members के size का जोड़ | सबसे बड़े member के बराबर |
| Valid members | एक साथ सभी valid | एक समय में सिर्फ एक valid |



Strings in C++ (C++ में स्ट्रिंग्स)

C++ में **string** एक टेक्स्ट (text) को represent करने के लिए इस्तेमाल होता है, यानी अक्षरों का collection होता है।

C++ में strings को दो तरीकों से represent किया जा सकता है:



1. C-style Strings (Character Arrays)

यह पुराना तरीका है जिसमें string को character array के रूप में represent किया जाता है और अंत में '\0' (null character) होता है।

Example:

```
#include <iostream>  
using namespace std;  
  
int main() {  
    char name[10] = "Hello";  
    cout << "C-style string: " << name << endl;  
    return 0;  
}
```

C-style string का इस्तेमाल करते समय हमें size और null terminator का ध्यान रखना पड़ता है।

2. C++ Strings (string class)

यह नया और आसान तरीका है जिसमें हम C++ की string class का उपयोग करते हैं (जिसे <string> हेडर से include किया जाता है)। इसमें कई built-in functions भी होते हैं।

Example:

```
#include <iostream>  
#include <string> // Required for string  
using namespace std;  
  
int main() {  
    string name = "Hello";  
    cout << "C++ string: " << name << endl;  
  
    // String operations  
    cout << "Length: " << name.length() << endl;  
    name += " World"; // Concatenation  
    cout << "After concatenation: " << name << endl;  
  
    return 0;}  
}
```

C++ string class के basic functions

| Function Name | Definition / Work | Example Code | Output / Result |
|------------------------------------|---|--|-----------------|
| length() / size() | string की length return करता है | string s = "Hello"; cout << s.length(); | 5 |
| append() | string के अंत में text जोड़ता है | string s = "Good"; s.append(" Morning"); cout << s; | Good Morning |
| + (concatenation) | दो strings को जोड़ने के लिए उपयोग होता है | string s1 = "Hi", s2 = "There"; cout << s1 + " " + s2; | Hi There |
| at(index) | किसी index पर character return करता है | string s = "Apple"; cout << s.at(2); | p |
| substr(pos, len) | substring return करता है (position और length देकर) | string s = "Computer"; cout << s.substr(0, 4); | Comp |
| find("text") | substring की पहली position return करता है | string s = "Laptop"; cout << s.find("top"); | 3 |
| compare() | दो strings की तुलना (length) करता है (0 = equal, >0 या <0) | string a = "abc", b = "abc"; cout << a.compare(b); | 0 |
| empty() | चेक करता है कि string खाली है या नहीं (1 = true, 0 = false) | string s = ""; cout << s.empty(); | 1 |
| clear() | string को पूरी तरह से खाली कर देता है | string s = "Data"; s.clear(); cout << s; | (empty output) |
| strcpy() (C-style) | एक character array से दूसरे में पूरा string copy करता है | char s1[20]; strcpy(s1, "Hello"); cout << s1; | Hello |

3. Input String from User (C++ में यूज़र से string इनपुट लेना)

C++ में यूज़र से string इनपुट लेने के दो तरीके होते हैं:

Using cin (single word input)

cin का उपयोग तब किया जाता है जब हमें एक ही शब्द (word) इनपुट लेना हो।

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string name;
    cout << "Enter your name: ";
    cin >> name; // यह space के बाद का हिस्सा ignore करता है
    cout << "You entered: " << name << endl;
    return 0;
}
```

Note : अगर यूज़र "Rahul Sharma" टाइप करता है, तो सिर्फ Rahul ही लिया जाएगा।

Using getline() (multiple words input)

अगर हम पूरा वाक्य या नाम (जिसमें spaces हों) लेना चाहते हैं, तो getline() का उपयोग करते हैं।

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string name;
    cout << "Enter your full name: ";
    getline(cin, name); // पूरा इनपुट line के रूप में लेता है
    cout << "You entered: " << name << endl;
    return 0;
}
```



File Handling in C++ (फाइल हैंडलिंग)

C++ में **file handling** का उपयोग फ़ाइलों से **data पढ़ने (read)**, **लिखने (write)** और उन्हें **manage** करने के लिए किया जाता है। इसके लिए `<fstream>` header file का उपयोग किया जाता है। इसमें तीन मुख्य classes होती हैं:

| Class | Work |
|-----------------|-------------------------------------|
| ifstream | फाइल से data पढ़ने के लिए (Input) |
| ofstream | फाइल में data लिखने के लिए (Output) |
| fstream | पढ़ने और लिखने दोनों के लिए |

C++ program for File Read and Write :-

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main () {
ofstream MyWriteFile("filename.txt"); // Create a text file
MyWriteFile << "Hello all KTBU Students"; // Write to the file
MyWriteFile.close();
string myText;
ifstream MyReadFile("filename.txt");
while (getline (MyReadFile, myText)) {
cout << myText;
}
MyReadFile.close();
}
```

Output:

```
Hello all KTBU Students
```

Explanation :

`ofstream MyWriteFile("filename.txt");` → नई फाइल बनती है या पुरानी खुलती है और हम उसमें लिखते हैं।

`MyWriteFile << "Hello...";` → फाइल में टेक्स्ट लिखा जाता है।

`MyWriteFile.close();` → फाइल को बंद कर दिया जाता है।

`ifstream MyReadFile("filename.txt");` → फाइल को पढ़ने के लिए खोला जाता है।

`getline(MyReadFile, myText)` → फाइल से line-by-line text पढ़ा जाता है।

`cout << myText;` → स्क्रीन पर कंटेंट दिखाया जाता है।

1. Program to Write to a File

```
#include <iostream>
#include <fstream>
using namespace std;
```

```

int main() {
    ofstream MyFile("data.txt");
    MyFile << "Hello KTBU Students!";
    MyFile.close();
    cout << "Data written to file successfully.";
    return 0;
}

```

Output (Screen):

Data written to file successfully.

File Content (data.txt):

Hello KTBU Students!

Linear Search

परिचय:

लिनियर सर्च एक प्रकार की **क्रमिक खोज (sequential search)** विधि है। इस विधि में, हम दिए गए इनपुट एरे (array) के हर एक तत्व को एक-एक करके चेक करते हैं और उसे उस कुंजी तत्व (key element) से मिलाते हैं जिसे हमें ढूँढना है।

उदाहरण:

अगर हमें किसी एरे में **33** को खोजना है, तो सर्च पहले इंडेक्स (0th index) से शुरू होगा और एक-एक करके सभी तत्वों की तुलना 33 से करेगा। जब उसे 33 मिलेगा, तो खोज सफल होगी।

अगर हम एरे में **46** को खोजना चाहें और वह मौजूद न हो, तो पूरी एरे चेक करने के बाद भी कोई मेल नहीं मिलेगा और यह एक असफल खोज होगी।

लिनियर सर्च का एल्गोरिदम (Algorithm):

Step 1:

इनपुट एरे के **0**वें इंडेक्स से शुरू करें और कुंजी (key) को उस तत्व से तुलना करें।

Step 2:

अगर तत्व कुंजी से मेल खाता है, तो उस स्थान (इंडेक्स) को **return** कर दें।

Step 3:

अगर मेल नहीं खाता, तो अगले तत्व से तुलना करें।

Step 4:

Step 3 को तब तक दोहराएं जब तक कोई मेल नहीं मिल जाता।

Step 5:

अगर पूरा एरे चेक करने के बाद भी कोई मेल नहीं मिले, तो बताएं कि तत्व मौजूद नहीं है और प्रोग्राम को समाप्त करें।

बाइनरी सर्च

Definition: बाइनरी सर्च एक कुशल (efficient) खोज विधि है, जिसका उपयोग सॉर्ट किए गए (sorted) एवं किया जाता है। यह खोज विधि बार-बार एवं को दो भागों में विभाजित (divide) करती है और मध्य (middle) तत्व के आधार पर यह तय करती है कि अगली खोज बाएं भाग में होगी या दाएं भाग में।

ध्यान दें: बाइनरी सर्च केवल तभी काम करता है जब एवं पहले से सॉर्ट (क्रमबद्ध) हो।

कार्य प्रणाली (Working Process):

- मध्य तत्व (middle element) को चुनें।
- अगर मध्य तत्व ही key है, तो खोज सफल।
- अगर key < मध्य तत्व, तो बाएं भाग (left half) में खोज करें।
- अगर key > मध्य तत्व, तो दाएं भाग (right half) में खोज करें।
- उपरोक्त प्रक्रिया को तब तक दोहराएं जब तक तत्व मिल न जाए या खोज क्षेत्र खाली हो जाए।

उदाहरण (Example):

मान लीजिए कि हमारे पास यह सॉर्ट किया हुआ एवं है:

[11, 22, 33, 44, 55, 66, 77]

हमें 33 को खोजना है।

- Step 1: Middle = 44 (index 3), 33 < 44 → Left half देखें।
- Step 2: New array = [11, 22, 33], Middle = 22 → 33 > 22 → Right half देखें।
- Step 3: New array = [33], Middle = 33 → मिल गया! ✓

बाइनरी सर्च एल्गोरिदम (Binary Search Algorithm):

step 1: low = 0, high = n-1 (जहाँ n एवं की लंबाई है)

$$= \text{mid} = \frac{0+6}{2} = 3$$

step 2: mid = (low + high) / 2

step 3:

- अगर array[mid] == key → सफलता (return index)
- अगर key < array[mid] → high = mid - 1
- अगर key > array[mid] → low = mid + 1

$$\begin{array}{l} \text{mid} = 3 \quad - \quad \times \\ \text{low} = 0 \quad , \quad \text{high} = 6 \\ \text{low} = 0 \quad \quad \quad \text{high} = 2 \end{array}$$

step 4:

- जब तक low <= high, तब तक step 2 और 3 दोहराएं।

step 5:

- अगर नहीं मिला तो बताएँ: "तत्व मौजूद नहीं है।"

मुख्य बिंदु (Key Points):

- काम करता है सिर्फ sorted array पर।

समय जटिलता (Time Complexity):

→ Best Case: O(1)

→ Worst Case: O(log n)

स्पेस जटिलता (Space Complexity): O(1) (iterative method में)

Example

$$\text{Ans} = 33$$

| | | | | | | | |
|------|----|----|----|----|----|----|----|
| a[7] | 11 | 22 | 33 | 44 | 55 | 66 | 77 |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

$$\text{low} = 0, \quad \text{high} = 6 \quad \text{mid} = \frac{\text{low} + \text{high}}{2} = \frac{0+6}{2} = 33$$

$$a[3] == 33 / 44 == 33 \times$$

$$\text{low} = 0, \quad \text{high} = (\text{mid}-1) = 3-1 = \text{high}=2$$

| | | |
|----|----|----|
| 11 | 22 | 33 |
| 0 | 1 | 2 |

$$\text{mid} = \frac{\text{low} + \text{high}}{2} = \frac{0+2}{2}$$

$$\text{mid} = 1$$

$$\text{if } a[1] == 33 / 22 == 33 \times$$

$$\text{low} = \text{mid}+1=1+1=2, \quad \text{high} = 2$$

| |
|----|
| 33 |
| 2 |

$$\text{mid} = \frac{\text{low} + \text{high}}{2} = \frac{2+2}{2} = 2$$

$$\text{mid} = 2 \quad a[2] == 33 \quad | \quad 33 == 33$$

WITB