সহজ ভাষায় প্রোগ্রামিং সি/সি++

আহমদ ওয়াহিদ মাসুদ

সি ল্যাঙ্গুয়েজের বিভিন্ন গুরুত্বপূর্ণ ফিচারের একটি হলো পয়েন্টার। পয়েন্টার নিয়ে গত কয়েক পর্ব ধরে আলোচনা করা হচ্ছে। পয়েন্টার ব্যবহার করা যেমন একই সাথে কিছুটা কঠিন, তেমনি এর ব্যবহার প্রোগ্রামারকে অনেকটাই সহজ করে দেয়। এ ছাড়া পয়েন্টার দিয়ে এমন অনেক কাজই করা সম্ভব যা, অন্য কোনোভাবে করা সম্ভব নয়। পয়েন্টারের কাজ অনেক ধরনের হলেও এর সাথে সি ল্যাঙ্গুয়েজের একটি ফিচারের বেশ মিল দেখা যায়– তা হলো অ্যারে। পয়েন্টারের বেসিক ধারণা এবং অ্যারের বেসিক ধারণার মাঝে অনেক মিলই দেখা যায়। আর তাই পয়েন্টারকে চাইলে অ্যারের মতো, আবার অ্যারেকে চাইলে মাঝেমধ্যে পয়েন্টারের মতো ব্যবহার করা যায়। তবে মনে রাখতে হবে, অবশ্যই পয়েন্টার এবং অ্যারের উদ্দেশ্য ভিন্ন এবং এদের মাঝে কিছুটা মিল থাকলেও মূল কাজ সম্পূর্ণ আলাদা। এ পর্বে পয়েন্টার এবং অ্যারের মাঝে যোগসূত্র নিয়ে আলোচনা করা হয়েছে।

এখানে যেহেতু পয়েন্টার এবং অ্যারে দুটি বিষয় নিয়ে আলোচনা করা হচ্ছে, তাই এদের সম্পর্কে অবশ্যই প্রাথমিক ধারণা থাকতে হবে। পয়েন্টার নিয়ে যেহেতু গত কয়েক পর্বে আলোচনা করা হয়েছে, তাই এখন আর এর প্রাথমিক ধারণা নিয়ে আলোচনা করা হবে না। তবে অ্যারে সম্পর্কে মনে করিয়ে দেয়ার জন্য সংক্ষেপে আলোচনা করা হয়েছে। অ্যারে হলো অনেকগুলো ভেরিয়েবল একসাথে ডিক্লেয়ার করার একটি পদ্ধতি। ধরা যাক, কোনো প্রোগ্রামে একই সাথে পাঁচটি ভেরিয়েবল ডিক্লেয়ার করার প্রয়োজন হলো। তাহলে ইউজার সাধারণ নিয়মে পাঁচটি ভেরিয়েবল ডিক্লেয়ার করতে পারেন। এজন্য পাঁচটি স্টেটমেন্ট লেখার প্রয়োজন হবে। আবার একটি স্টেটমেন্টেও পাঁচটি ভেরিয়েবল ডিক্লেয়ার করা যায়, তবে সে ক্ষেত্রে স্টেটমেন্টটি অনেক বড় হবে। কিন্তু অ্যারে ব্যবহার করে পাঁচটি ভেরিয়েবল একই সাথে অর্থাৎ একটি স্টেটমেন্ট দিয়েই ডিক্লেয়ার করা সম্ভব। অ্যারে ডিক্লেয়ার করার নিয়ম : data_type array_name[array_size]। অ্যারের নাম হলো যেকোনো ভেরিয়েবলের নাম। অর্থাৎ ভেরিয়েবলের নামের নিয়মানুসারে অ্যারের নামকরণ করতে হবে। এখানে নতুন বিষয় হলো অ্যারে সাইজ। যেহেতু অ্যারে হলো অনেকগুলো ভেরিয়েবলের সমষ্টি। সুতরাং কতগুলো ভেরিয়েবল নিয়ে অ্যারে ডিক্লেয়ার করা হচ্ছে সেটা বলতে হবে। এটিই হলো অ্যারে সাইজ। সাইজ যত হবে, ততগুলো ভেরিয়েবল নিয়ে অ্যারে গঠিত হবে। অ্যারে এবং ভেরিয়েবল ডিক্লেয়ারের মাঝে মূল পার্থক্য হলো সাইজ। যেমন : int roll[10], class[5]; ইত্যাদি। এখানে একই সাথে রোল নামে ১০টি, ক্লাস নামে পাঁচটি ইন্টিজার ডিক্লেয়ার করা হয়েছে। আর রোল নামের যে ১০টি ভেরিয়েবল ডিক্লেয়ার করা হয়েছে, এগুলো আলাদাভাবেও অ্যাক্সেস করা যাবে। সে ক্ষেত্রে roll[0], roll[1], roll[2] ইত্যাদি হবে একেকটি ভেরিয়েবলের নাম।

এখন দেখা যাক, অ্যারে এবং পয়েন্টারের মাঝে কেমন মিল আছে। সি ল্যাঙ্গুয়েজে অ্যারে এলিমেন্টকে পয়েন্টার নোটেশনের মাধ্যমেও ব্যবহার করা যায়। আমরা জানি, অ্যারে ব্যবহার করার জন্য প্রথমে তা ডিক্লেয়ার করতে হয়। যেমন উপরে রোল নামের যে অ্যারে ডিক্লেয়ার করা হয়েছে সেখানে ১০টি ভেরিয়েবল মূলত ডিক্লেয়ার হয়ে গেছে। কিন্তু অ্যারে ডিক্লেয়ার করলে তার জন্য একটি বেস অ্যাড্রেস নির্ধারিত হয়ে যায়। সম্পূর্ণ অ্যারের জন্যই ওই বেস অ্যাড্রেসটি কাজ করে। ধরা যাক, রোল নামের অ্যারেটি মেমরির ০০০৩ অ্যাড্রেস থেকে শুরু হলো। অর্থাৎ অ্যারেটির বেস অ্যাড্রেস হলো ০০০১। আসলে অ্যারে ডিক্লেয়ার করলে শুধু এই বেস অ্যাড্রেসটিই নির্ধারিত হয়, আর কিছু হয় না। এর মানে হচ্ছে রোল [০]-এর মান বসবে ০০০১-০০০২ অ্যাড্রেসে। রোল [১]-এর মান বসবে ০০০৩-০০০৪ অ্যাড্রেসে। এভাবে ইউজার সর্বমোট ১০টি ভেরিয়েবল ব্যবহার করতে পারবেন। এখানে লক্ষণীয়, এই ০০০১ অ্যারেটির বেস অ্যাড্রেস, এটি শুধু রোল অ্যারের জন্যই কাজ করবে। অন্য কোনো অ্যারে ডিক্লেয়ার করলে তার বেস অ্যাড্রেস আলাদা হবে।

এখন আসা যাক, অ্যারের এ বিষয়টি কীভাবে পয়েন্টারের সাথে সম্পর্কিত। অ্যারের প্রথম এলিমেন্টের অ্যাড্রেস জানলে পরের এলিমেন্টের অ্যাড্রেসও ইনক্রিমেন্ট করে বের করা যায়। যেমন, অ্যারের প্রথম এলিমেন্টের অ্যাড্রেস হলো ০০০১। সুতরাং দ্বিতীয় এলিমেন্টের অ্যাড্রেস হবে ০০০৩। এখানে দুই করে ইনক্রিমেন্ট হচ্ছে, কারণ এটি ইন্টিজার টাইপের অ্যারে এবং আমরা জানি একটি ইন্টিজার ভেরিয়েবল ২ বাইট করে জায়গা নেয়। তাই এটি যদি একটি ক্যারেক্টার টাইপের অ্যারে হতো তাহলে এর অ্যাড্রেস ১ করে ইনক্রিমেন্ট হতো। এখন অ্যারের প্রথম এলিমেন্টের অ্যাড্রেস যদি ইউজার জানেন, তাহলে পরের এলিমেন্টের অ্যাড্রেসও ইউজার সহজেই ইনক্রিমেন্ট করে বের করতে পারবেন। ধরা যাক, প্রথম এলিমেন্টের অ্যাড্রেস অর্থাৎ ০০০১-কে একটি পয়েন্টারের মাঝে রাখা হলো। তাহলে ওই পয়েন্টারটিকে ইনক্রিমেন্ট করলেই পরের এলিমেন্টগুলো পাওয়া যাবে। ব্যাপারটি বোঝানোর জন্য একটি ছোট প্রোগ্রাম উদাহরণ হিসেবে দেয়া হলো।

```
int main
{
        int i=0, roll[5]={1,2,3,4,5};
        int *x;
        x=&roll[0];
        for(i=0;i<5;i++)
        {
                printf(“%d”,x);
                x=x+2;
        }
}
return 0;
```

এখানে প্রথমে একটি ইন্টিজার ভেরিয়েবল নেয়া হয়েছে পাঁচবার একটি লুপ চালানোর জন্য। এরপর রোল নামের একটি অ্যারে ডিক্লেয়ার করা হয়েছে এবং একই সাথে তার মান অ্যাসাইন করে দেয়া হয়েছে। এখন x নামে একটি ইন্টিজার টাইপের পয়েন্টার ডিক্লেয়ার করা হলো। এই পয়েন্টারটির কাজ হলো অ্যারের বিভিন্ন এলিমেন্টকে পয়েন্ট করে ব্যবহার করা। তার ডিক্লেয়ার করার পরপরই পয়েন্টারের মান হিসেবে অ্যারের প্রথম এলিমেন্টের অ্যাড্রেসকে নির্ধারণ করা হয়েছে। সুতরাং এখন যদি পয়েন্টারকে ব্যবহার করে প্রিন্ট করার নির্দেশ দেয়া হয়, তাহলে রোল [০]-এর মান বা অ্যারের প্রথম এলিমেন্টের মান প্রিন্ট হবে। আবার রোল অ্যারেটি যেহেতু ইন্টিজার টাইপের, তাই এর প্রতিটি এলিমেন্টের অ্যাড্রেস ২ করে ইনক্রিমেন্ট হবে। তাই একটি লুপের মাঝে পয়েন্টারের মান প্রিন্ট করার পর তার মান ২ করে ইনক্রিমেন্ট করা হয়েছে। এভাবে পয়েন্টার ব্যবহার করে অ্যারের সব ভেরিয়েবল প্রিন্ট করা হলো। এটাকেই বলে পয়েন্টার নোটেশন দিয়ে অ্যারে ব্যবহার করা।

ইউজার প্রশ্ন করতে পারেন, পয়েন্টার নোটেশন দিয়ে অ্যারে ব্যবহার করার কী দরকার। সরাসরি অ্যারে ব্যবহার করলেই তো আর কোনো ঝামেলা থাকে না। আসলে ওপরের উদাহরণের মতো ছোট প্রোগ্রামে কোনো পার্থক্যই বোঝা যাবে না। কারণ পয়েন্টার নোটেশন ব্যবহারের সুবিধাটা আসলে রান টাইমে। বেশিরভাগ কমপিউটারের আর্কিটেকচার ওপরের উদাহরণের মতো পয়েন্টার নোটেশনে দ্রুত কাজ করবে। কমপিউটারের হিসাব করার পদ্ধতিটিই এভাবে ডিজাইন করা হয়েছে। বিষয়টি একটু বিস্তারিতভাবে বলা যাক। অ্যারের প্রতিটি এলিমেন্টকে অ্যাক্সেস করার সাধারণ নিয়ম হলো তার বেস অ্যাড্রেস থেকে ওই এলিমেন্টের অ্যাড্রেসে যাওয়া। যেমন, রোল [০]-এর অ্যাড্রেস বা অ্যারেটির বেজ অ্যাড্রেস হলো ০০০১। এখন প্রোগ্রামকে যদি বলা হয় রোল ▶

পয়েন্টার নোটেশন এবং মাল্টি ডাইমেনশনাল অ্যারে

অ্যারের ক্ষেত্রে মাল্টি ডাইমেনশনের কথা শুনলেই প্রোগ্রামারের মনে একটু ভয় ঢোকে। কারণ মাল্টি ডাইমেনশনের অ্যারে ব্যবহার খুবই কঠিন কাজগুলোর একটি। একই সাথে প্রতিটি ডাইমেনশনের হিসাব রাখতে গেলে বেশিরভাগ সময়ই ওলটপালট হয়ে যায়। আর একবার ডাইমেনশনের হিসাবে ভুল হয়ে গেলে তা ঠিক করা খুবই কঠিন। কিন্তু মজার ব্যাপার, পয়েন্টার নোটেশন দিয়ে মাল্টি ডাইমেনশন অ্যারেও ব্যবহার করা যায় এবং এ ক্ষেত্রে সাধারণ অ্যারের মতোই কাজটি সহজ হয়, শুধু সূত্রটি অন্য ধরনের হয়। মাল্টি ডাইমেনশন অ্যারে সম্পর্কে আলোচনার আগে একটু সংক্ষেপে ধারণা দেয়া হলো।

অ্যারের ক্ষেত্রে ডাইমেনশনের ধারণা কীভাবে আসে তা প্রথমে শুনতে একটু অদ্ভুত লাগতে পারে। তবে ইউজারের জেনে রাখা ভালো, অ্যারের মাঝেও ডাইমেনশনের কিছু ব্যাপার আছে। ডাইমেনশনের ব্যবহার আর কিছুই নয়, অ্যারেকে আরও প্রসারিত করা। অর্থাৎ সাধারণ অ্যারের জন্য যেখানে একটি এলিমেন্ট একটি ভেরিয়েবলের সমতুল্য, সেখানে একটি টুডি অ্যারের ক্ষেত্রে একটি এলিমেন্ট দুটি ভেরিয়েবলের সমান। সহজ কথায়, অ্যারের ডাইমেনশন যত বেশি হবে, অ্যারের প্রতিটি এলিমেন্ট দিয়ে ততগুলো ভেরিয়েবল প্রকাশ করা যাবে। যদিও অ্যারের ক্ষেত্রে মাল্টি ডাইমেনশন ব্যবহার করা যায়, তবুও প্রোগ্রামাররা সাধারণত টুডি ডাইমেনশন বেশি ব্যবহার করেন না। টুডি অ্যারে দিয়েই সব ধরনের কাজ হয়ে যায়, তাই মাল্টি ডাইমেনশনের আলোচনায় সাধারণত টুডি অ্যারের আলোচনাই করা হয়। তবে ইউজার চাইলে তার ইচ্ছে মতো ডাইমেনশন ব্যবহার করতে পারেন। সেটি থ্রিডি হোক কিংবা ফোরডি হোক। সাধারণত ম্যাট্রিক্সের অসুবিধার জন্য টুডি অ্যারে ব্যবহার করা হয়। টুডি অ্যারেকে রো এবং কলাম অনুযায়ী ভাগ করা যায়। তবে মেমরিতে কিন্তু এভাবে রো কলাম হিসেবে অ্যারে থাকে না। মেমরিতে অ্যারের গঠন সবসময় লিনিয়ার হয়। প্রোগ্রামে টুডি অ্যারে ব্যবহারের নিয়ম হলো :

```
data_type array_name[row_size][column_size];
```

রো সাইজ এবং কলাম সাইজ বলতে অ্যারের ইনডেক্স বোঝানো হচ্ছে। অর্থাৎ টুডি অ্যারের ক্ষেত্রে ইনডেক্স দুটি। যেমন, int roll[2][2]; এখানে একটি টুডি অ্যারে ডিক্লেয়ার করা হয়েছে, যার ইনডেক্স ২, ২। অর্থাৎ এর রো এবং কলাম দুটি করে। এখন আমরা একে একটি ম্যাট্রিক্স বা ছক আকারে কল্পনা করতে পারি। আর বোঝার সুবিধার্থে এভাবে কল্পনা করা হয়। কিন্তু মেমরিতে অ্যারেটির জন্য পরপর চারটি ভেরিয়েবল ডিক্লেয়ার হবে। সাধারণ অ্যারে ব্যবহারের সময় দেখানো হয়েছে ইনডেক্সিং ০ থেকে শুরু করা হয়। এ ক্ষেত্রেও তার ব্যতিক্রম নয়। তবে এখানে ডান দিকের বন্ধনীর মান প্রথমে বাড়ে, পরে বাম দিকের বন্ধনীর মান বাড়ে। অর্থাৎ প্রথমে কলাম, তারপর রো-এর মান বাড়ে। এখন অ্যারের কোনো নির্দিষ্ট এলিমেন্ট ব্যবহার করতে হলে সাধারণ অ্যারের নিয়ম মেনে চললেই হবে। শুধু পার্থক্য হলো এখানে একই সাথে দুটি ইনডেক্স ব্যবহার করতে হবে। যেমন, রোল অ্যারের প্রথম এলিমেন্টটি হলো roll[0][0], দ্বিতীয়টি হলো roll[0][1], তৃতীয়টি হলো roll[1][0] ইত্যাদি। আর ডিক্লেয়ার করার সময় দুইভাবে মান সরাসরি অ্যাসাইন করা যায়। যেমন, roll[2][2]={1,2,3,4} অথবা roll[2][2]={{1,2},{3,4}}। তবে দ্বিতীয় নিয়মটি শুধু টুডি অ্যারের জন্যই প্রযোজ্য।

টুডি অ্যারের কাজ সাধারণত বিভিন্ন গাণিতিক সমস্যাতে দেখা যায়। যেমন ম্যাট্রিক্স। অথবা হাই লেভেলের অ্যালগরিদমে টুডি অ্যারে ব্যবহার করতে হয়। এখন আসল কথা হলো টুডি অ্যারে কেনো সবসময় কঠিন এবং কীভাবে পয়েন্টার দিয়ে এর ব্যবহার সহজ করা যায়। সাধারণত টুডি অ্যারে যেসব প্রোগ্রামে অথবা অ্যালগরিদমে ব্যবহার করা হয়, সেই অ্যালগরিদমগুলোই অনেক কঠিন থাকে। অন্যান্য বিষয়ের সাথে টুডি অ্যারের ডাইমেনশনের হিসাব রাখাটা প্রোগ্রামারের জন্য খুবই কঠিন হয়ে যায়। আর সমস্যা যত কঠিন হয়, তাতে ভুল করার সম্ভাবনা তত বেশি হয় এবং টুডি অ্যারেতে একবার ভুল করলে সেটা মেলানো খুব বেশি কঠিন হয়ে যায়। তাই প্রোগ্রামাররা সাধারণত মাল্টি ডাইমেনশনের অ্যারের ব্যবহার এড়িয়ে চলতে চান। কিন্তু টুডি অ্যারে দিয়ে এমন অনেক সমস্যাই সহজে সমাধান করা সম্ভব, যা এমনকি অন্য কোনোভাবে সমাধান করা যায় না। তাই এটি ব্যবহার করা ছাড়া অন্য উপায়ও থাকে না। এ কারণেই টুডি অ্যারেকে যদি পয়েন্টারের নোটেশনের সাহায্যে ব্যবহার করা হয়, তাহলে প্রোগ্রামারের জন্য তা অনেক সহজ হয়ে যায়। পয়েন্টারের নোটেশনের সাধারণ সূত্রটি নিচে দেয়া হলো :

```
x[row][column]=*(x[row]+column);
```

একটি ছকের মাধ্যমে উদাহরণ দেয়া হলো :

ELEMENT	POINTER NOTATION
x[0][0]	*(x[0]+0)
x[0][1]	*(x[0]+1)
x[1][0]	*(x[1]+0)
x[1][1]	*(x[1]+1)

এভাবে সাধারণ অ্যারের মতো টুডি অ্যারেতেও পয়েন্টারের নোটেশন ব্যবহার করা যাবে। এতে একই সাথে যেমন প্রোগ্রামারের জন্য হিসাব রাখা সহজ হবে, তেমনি প্রোগ্রামের রানটাইমও কম হবে।

যদিও অ্যারে এবং পয়েন্টার দুটি সম্পূর্ণ ভিন্ন বিষয়, কিন্তু একটির মাধ্যমে আরেকটির ব্যবহার অনেক সহজ হয়ে যায়। এ ধরনের ব্যবহার সাধারণত জটিল প্রোগ্রামে অথবা অ্যাডভান্সড অ্যালগরিদমে দরকার হয়।

---

[০]-এর মান প্রিন্ট কর, তাহলে সেটি কোনো সমস্যা নয়। কারণ প্রোগ্রাম জানে বেজ অ্যাড্রেসটি কী। এখন যদি প্রোগ্রামকে বলা হয় রোল [৩]-এর মান প্রিন্ট কর, তাহলে প্রোগ্রাম আর আগের মতো সহজে কাজ করতে পারবে না। আগে প্রোগ্রাম জানত বেজ অ্যাড্রেসটিই হলো রোল [০]-এর অ্যাড্রেস। কিন্তু এবার প্রোগ্রাম জানে না রোল [৩]-এর অ্যাড্রেস কত। এটি বের করার জন্য প্রোগ্রামকে রোল [০] থেকে একটি একটি করে এলিমেন্ট পার হয়ে রোল [৩]-এ আসতে হবে, যেটি সময়সাপেক্ষ এবং এতে রানটাইম বেশি লাগে। কিন্তু পয়েন্টার নোটেশন দিয়ে অ্যারেকে অ্যাক্সেস করলে প্রোগ্রামকে আর শুরু থেকে অর্থাৎ রোল [০] সবগুলো ভ্রমণ করে আসতে হবে না, সরাসরি রোল [০]-এর অ্যাড্রেসের সাথে ৩*২ যোগ করলেই রোল [৩]-এর অ্যাড্রেস পাওয়া যাবে এবং প্রোগ্রাম সরাসরি রোল [৩]-এ অ্যাক্সেস করতে পারবে। এখানে ৩*২ যোগ করাটা একটি সূত্র অনুসরণ করে করা হয়েছে। এখানে ৩ হলো যে এলিমেন্টকে অ্যাক্সেস করতে হবে তার ইনডেক্স নাম্বার, আর ২ দেয়ার কারণ হলো এগুলো সব ইন্টিজার টাইপের ভেরিয়েবল। এগুলো ক্যারেক্টার টাইপের হলে ৩*১ যোগ করতে হতো। একটি ছোট প্রোগ্রামের জন্য এ সময়টুকু তেমন বিবেচনার বিষয় নয়। তবে বড় প্রোগ্রামের ক্ষেত্রে এই মেথড ব্যবহার করলে রানটাইম অনেক কমে যাবে। এ ধরনের মেথড ব্যবহার করলেই সফটওয়্যার লোড করতে বা চালাতে কম সময় লাগে। আর এ কারণেই এ ধরনের অ্যালগরিদমকে উন্নতমানের হিসেবে বিবেচনা করা হয়। কজ

ফিডব্যাক : wahid_cseaust@yahoo.com