সহজ ভাষায় প্রোগ্রামিং সি/সি++

আহমদ ওয়াহিদ মাসুদ

অ্যাডভান্সড সি প্রোগ্রামিংয়ের একটি অন্যতম উপাদান হলো পয়েন্টার ভেরিয়েবল। পয়েন্টার নিয়ে ইতোপূর্বে আলোচনা করা হয়েছে। পয়েন্টার কী, কেন এটি ব্যবহার করা হয় এবং কিভাবে এর মান নির্ধারণ করা হয় ইত্যাদি জানার পর বাকি থাকে পয়েন্টার ডাটা কিভাবে ব্যবহার করা যায়। এ লেখায় কিভাবে পয়েন্টার ডাটা ব্যবহার করা যায় এবং এর ফলে প্রোগ্রাম কিভাবে সহজ ও আকারে ছোট হয় তা নিয়ে বিস্তারিত আলোচনা করা হয়েছে।

পয়েন্টেড ডাটা নিয়ে কাজ

পয়েন্টারের জন্য অন্য যে ভেরিয়েবলের মেমরি অ্যাড্রেস নির্ধারণ করা হয়, সেই ভেরিয়েবলের ডাটা অর্থাৎ পয়েন্টেড ডাটা প্রোগ্রামে বিভিন্নভাবে ব্যবহার করা যায়। পয়েন্টেড ডাটা ব্যবহারের বিশেষ সুবিধা হলো, এতে মূল ভেরিয়েবলের ডাটা অপরিবর্তিত রাখা সম্ভব। তবে ইউজার ইচ্ছে করলে পয়েন্টারের মাধ্যমে মূল ভেরিয়েবলের ডাটাও পরিবর্তন করতে পারেন। সে ক্ষেত্রে আসলে পয়েন্টারের ব্যবহারের কোনো প্রয়োজন থাকে না। তাছাড়া পয়েন্টারের ব্যবহারের মাধ্যমে মেমরি তুলনামূলক কম ব্যবহার হয়। কারণ একটি ডাবল টাইপ ভেরিয়েবল ডিক্লেয়ার করা হলে সাধারণত মেমরির ৮ বাইট জায়গা অ্যালোকেট হয়। কিন্তু একটি ডাবল টাইপ পয়েন্টার ডিক্লেয়ার করলে মূল ডাবল ভেরিয়েবলের অ্যাড্রেস রাখার জন্য যতটুকু জায়গা রাখা প্রয়োজন শুধু ততটুকুই মেমরিতে অ্যালোকেট হয় এবং সেটি সাধারণত ২/১ বাইটের বেশি হয় না। তবে এটি নির্ভর করে মেমরির সাইজ এবং তার অ্যাড্রেসের ওপর। তাই একটি বড় প্রোগ্রামে যখন অসংখ্য ভেরিয়েবল থাকে, তখন কোনো মূল ভেরিয়েবল অপরিবর্তিত রেখে তা ব্যবহারের জন্য পয়েন্টার ব্যবহার করা উচিত। যেমন– একটি প্রোগ্রামে যদি ১ হাজার ডাবল ভেরিয়েবল থাকে এবং তাদের ডাটা অপরিবর্তিত রেখে ব্যবহারের জন্য যদি আরও ১ হাজার ডাবল ভেরিয়েবল ডিক্লেয়ার করা হয় তাহলে মোট ২০০০×৮=১৬০০০ বাইট জায়গা দরকার। কিন্তু ১০০০টি পয়েন্টার ব্যবহার করলে ১০০০x৮+১০০০x২=১০০০০ বাইট জায়গা দরকার।

মূলত কোনো মেমরি অ্যাড্রেসে রাখা ডাটা নিয়ে কাজ করার জন্যই পয়েন্টার ব্যবহার করা হয়। আর পয়েন্টার নিয়ে কাজ করার জন্য ইনডিরেক্ট অপারেটর (*) এবং অ্যাড্রেস অপারেটর (&) ব্যবহার হয়। ইনডিরেক্ট অপারেটরের ব্যবহার দেখানোর জন্য একটি ছোট প্রোগ্রাম দেয়া হলো।

```
int x,y,*z;
x=44;
y=40;
z=&x;
printf(“x=%d\ty=%d\tz=%x”);
y=*z;
*z=10;
printf(“x=%d\ty=%d\tz=%x”);
```

এখানে দুটি ইন্টিজার এবং একটি ইন্টিজার পয়েন্টার ডিক্লেয়ার করা হয়েছে। ইন্টিজার দুটির মান ডিক্লেয়ার অ্যাসাইন করে দেয়া হয়েছে। আর পয়েন্টারে x ভেরিয়েবলের অ্যাড্রেস দেয়া হয়েছে। তাই প্রথমে প্রিন্টের সময় 44 40 fff8 প্রিন্ট হবে। প্রথম সংখ্যা দুটি হলো যথাক্রমে x এবং y-এর মান এবং পরের সংখ্যাটি একটি হেক্সাডেসিমাল সংখ্যা, যা পয়েন্টারটির অ্যাড্রেস। কমপিউটারের মেমরির লোকেশন সবসময় হেক্সাডেসিমালে থাকে। খেয়াল রাখতে হবে z একটি পয়েন্টার হলেও তার নিজেরও একটি অ্যাড্রেস আছে, যা এখানে প্রিন্ট হয়েছে। কিন্তু যেহেতু এ পয়েন্টারের মাঝে x রাখা হয়েছে, তার মানে মেমরির fff4 লোকেশনে 44 সংখ্যাটি আছে। আরেকটি গুরুত্বপূর্ণ বিষয় খেয়াল রাখতে হবে, ইন্টিজার সংখ্যা প্রিন্ট করতে হলে %d আর ফ্লোট/ডাবল প্রিন্ট করতে হলে %f ব্যবহার হয়। কিন্তু কোনো হেক্সাডেসিমাল সংখ্যা সরাসরি প্রিন্ট করার দরকার হলে %x ব্যবহার করতে হয়।

পয়েন্টারের ডাটা বোঝাতে (*) অপারেটর ব্যবহার হয়। তাই ïay z বললে fff8 বোঝাবে, কিন্তু *z বললে x-এর মান 44 বোঝাবে।

প্রথম প্রিন্টের পরে y-এর মান হিসেবে *z ব্যবহার করা হয়েছে। অর্থাৎ এখন x -এর মান y -এর মান হিসেবে নির্ধারণ হবে। আবার *z বা x-এর মান হিসেবে ১০ নির্ধারণ করা হয়েছে। তাই দ্বিতীয় প্রিন্টের সময় 10 44 fff8 প্রিন্ট হবে।

*z বলতে আসলে z-এর ভেতরে যে অ্যাড্রেস আছে তার কনটেন্ট বোঝায়। আর তাই y=*z করার জন্য y-এর মান হিসেবে x-এর মান নির্ধারণ হয়েছে। এভাবে ইনডিরেক্ট অপারেটরের মাধ্যমে পয়েন্টার দিয়ে অন্যান্য ভেরিয়েবলের ডাটা নিয়ন্ত্রণ করা যায়।

উপরের উদাহরণ দেখে মনে হতে পারে সি-তে পয়েন্টারের ব্যবহার একটি বাহুল্য ছাড়া আর কিছু নয়। কিন্তু আগেই বলা হয়েছে, পয়েন্টারের ব্যবহার সাধারণত অ্যাডভান্সড সি এবং অনেক বড় ও জটিল প্রোগ্রামের ক্ষেত্রে প্রয়োজন হয়। উপরের প্রোগ্রামটিতে পয়েন্টারের ব্যবহারের কোনো দরকার ছিল না। তবে এখানে শুধু পয়েন্টারের ব্যবহার সম্পর্কে একটি উদাহরণ দেয়া হলো। পয়েন্টারের অনেক গুরুত্বপূর্ণ এবং প্রচলিত ব্যবহারের মাঝে একটি হলো ডায়নামিক প্রোগ্রামিং। সাধারণ প্রোগ্রামিংয়ে একটি নির্দিষ্ট পরিমাণ জায়গা আগে থেকেই অ্যালোকেট করে দেয়া হয়। কিন্তু ডায়নামিক প্রোগামিংয়ে আগে থেকে কোনো জায়গা অ্যালোকেট করা হয় না। যেমন– একটি সাধারণ যোগ করার প্রোগ্রামে ১০০টি এলিমেন্টের একটি অ্যারে ডিক্লেয়ার করা হলো, যাতে ইউজার পরপর ১০০টি নাম্বার ইনপুট নিয়ে তা যোগ করতে পারেন। কিন্তু এ প্রোগ্রামের অনেক বড় একটি দুর্বলতা হলো তা অ্যারে ব্যবহার করে করা হয়েছে। কারণ, বাস্তবে যখন ইউজার প্রোগ্রামটি চালাবেন তখন যদি তার ১০০টির বেশি সংখ্যা যোগ করতে হয়, তাহলে প্রোগ্রামটি ঠিকমতো কাজ করবে না। আবার প্রোগ্রামটির শুরুতেই যদি ১০০ এলিমেন্টের একটি অ্যারে ডিক্লেয়ার করা হয়, তাহলে মেমরিতে তখনই ১০০ ভেরিয়েবলের একটি বিশাল জায়গা অ্যালোকেট হবে, যা অনেক সময় প্রোগ্রামটিকে হ্যাং করে দিতে পারে। এখন ইউজার মাত্র ৩টি সংখ্যা যোগ করলে বাকি ৯৭টি ভেরিয়েবল অব্যবহৃত অবস্থায় থেকে যাবে, অর্থাৎ মেমরি অপচয় হবে। এ ধরনের সমস্যা সমাধানের জন্য ডায়নামিক প্রোগ্রামিংয়ের প্রয়োজন, যেখানে ইউজারের যতগুলো সংখ্যা যোগ করার প্রয়োজন শুধু ততগুলো ভেরিয়েবলের অ্যারে ডিক্লেয়ার হবে, যাতে মেমরি অপচয় না হয়। আবার প্রোগ্রামের শুরুতেই কোনো মেমরি অ্যালোকেট হবে না, যখন ইনপুট নেয়া হবে তখনই শুধু স্বয়ংক্রিয়ভাবে ভেরিয়েবল মেমরিতে অ্যালোকেট হবে। এ ধরনের কাজ একমাত্র পয়েন্টারের সাহায্যেই করা সম্ভব। অ্যাডভান্সড প্রোগ্রামিংয়ের আরও অনেক ক্ষেত্র আছে, যেখানে পয়েন্টারের ব্যবহার খুব দরকার।

পয়েন্টার অ্যারিথমেটিক এবং এক্সপ্রেশন

অন্যান্য সাধারণ ভেরিয়েবলের মতো পয়েন্টার দিয়েও বিভিন্ন এক্সপ্রেশন তৈরি করা যায়। যেমন :

```
int i=3,j=5,*p=&i,*q=&j,*r;
double x;
```

ভেরিয়েবলগুলো ডিক্লেয়ার করার পর এক্সপ্রেশনগুলো একটি ছকে দেয়া হলো।

এক্সপ্রেশন	**ইভ্যালুয়েশনের অর্ডার**	**মান**
p==&i	p==(&i)	1
**&p	(*(*(&p)))	3
r=&x	r=(&x)	এরর
7**p/*q+7	(7*(*p))/((*q)+7)	1
(r=&j)=*p	(*(r=&j))*=(*p)	15

প্রথম একপ্রেশনটি একটি রিলেশনাল এক্সপ্রেশন, যা কিনা সত্য। তাই এর মান ১।

দ্বিতীয় স্টেটমেন্টে &p, q-এর অ্যাড্রেস দিচ্ছে, *(&p) মানে হলো p-এর কনটেন্ট এবং *(*(&p)) মানে হলো *(p-এর অ্যাড্রেস)। অর্থাৎ p-এর কনটেন্ট হিসেবে যে অ্যাড্রেস আছে তার কনটেন্ট। p-এর ভেতরে i-এর অ্যাড্রেস আছে। সুতরাং &p লেখা মানে হলো p-এর ভেতরে যে i-এর অ্যাড্রেস আছে তার কনটেন্ট। অর্থাৎ i-এর মান। কিন্তু &p বলতে p-এর নিজের অ্যাড্রেস বোঝায়। সুতরাং *(&p) বলতে ▸

p-এর নিজের কনটেন্ট বোঝায়, যা কিনা i-এর অ্যাড্রেস। আবার *(*(&p)) বলতে *(&p)-এর কনটেন্ট অর্থাৎ i-এর মান বোঝায়। তাই দ্বিতীয় স্টেটমেন্টের মান হলো i-এর মান অর্থাৎ ৩। তৃতীয় স্টেটমেন্টটি ভুল। কারণ r একটি ইন্টিজার টাইপের পয়েন্টার এবং x একটি ডাবল টাইপের ভেরিয়েবল। আমরা জানি, পয়েন্টার ভেরিয়েবল এবং পয়েন্টেড ভেরিয়েবল একই টাইপের হতে হয়। চতুর্থ স্টেটমেন্টের মান ১। এখানে লক্ষণীয় বিষয়, (*) অপারেটরটি ইনডিরেক্ট এবং মাল্টিপ্লাইয়ার দুটি হিসেবেই ব্যবহার হচ্ছে। প্রথমে 7-এর সাথে *p-এর গুণ হচ্ছে। এ কারণে 7**p লেখা হয়েছে। অর্থাৎ এখানে প্রথম (*) মাল্টিপ্লাই করছে এবং দ্বিতীয় (*) পয়েন্ট করছে। পঞ্চম লাইনে প্রথমে r-এর ভেতরে j-এর অ্যাড্রেস যাচ্ছে। তারপর বাম দিকের (*) দিয়ে অর্থাৎ *(r=&j) এটা দিয়ে j-এর মান তথা 5 বোঝাচ্ছে। তার সাথে *= সাইন ব্যবহার করে মাল্টিপ্লাই এবং ইক্যুয়াল করা হচ্ছে ডান দিকের (*p), অর্থাৎ i-এর মান, যা কিনা 3। সুতরাং 3*5 বা 15 হবে *r বা j-এর নতুন মান।

এভাবে পয়েন্টার ভেরিয়েবল ব্যবহার μ-এ বিভিন্ন ধরনের এক্সপ্রেশন তৈরি করা সম্ভব। কিন্তু খেয়াল রাখতে হবে, একটি পয়েন্টার ভেরিয়েবলকে অন্য পয়েন্টার ভেরিয়েবলের সাথে যোগ-বিয়োগ কিংবা অন্য কোনো সংখ্যা দিয়ে গুণ-ভাগ করা যাবে না। যেমন– *x,*y দুটি পয়েন্টার হলে x/y বা x+y করা যাবে না। এর ফলে অ্যাড্রেসের মাঝে যোগ-বিয়োগ করা হয়, যা সম্ভব নয়। তাই এ ক্ষেত্রে প্রোগ্রাম এরর দেখাবে। তবে একটি পয়েন্টারে রাখা অন্য কোনো ভেরিয়েবলের ডাটার সাথে কোনো সংখ্যা যোগ-বিয়োগ করা যায়। যেমন– int p[10]; x=p; x=x+2; এখানে প্রথমে একটি অ্যারে ডিক্লেয়ার করা হয়েছে। তারপর একটি পয়েন্টার দিয়ে অ্যারেটিকে পয়েন্ট করা হয়। অর্থাৎ দ্বিতীয় লাইন এক্সিকিউট হওয়ার পর x দিয়ে অ্যারেটির প্রথম এলিমেন্টকে ব্যবহার করা যাবে। আর শেষ লাইনটি এক্সিকিউট হওয়ার পর x পয়েন্টারটি অ্যারেটির তৃতীয় এলিমেন্টকে পয়েন্ট করবে। একইভাবে পয়েন্টার ভেরিয়েবলের সাথে ইনক্রিমেন্ট বা ডিক্রিমেন্ট অপারেটর ব্যবহার করেও অ্যারেথমেটিক এক্সপ্রেশন তৈরি করা যায় এবং সে ক্ষেত্রে প্রোগ্রাম কোনো এরর দেখাবে না। যেমন– উপরের উদাহরণে x=x+2 না লিখে ++x; লিখলে তখন x পয়েন্টারটি অ্যারেটির দ্বিতীয় এলিমেন্টকে পয়েন্ট করবে। (++) অপারেটরের কাজ না জানলে এ ধরনের এক্সপ্রেশন তৈরি করা যাবে না। তাই এখানে এ অপারেটর সম্পর্কে সংক্ষিপ্ত বর্ণনা দেয়া হলো।

++ অপারেটরকে ইনক্রিমেন্ট অপারেটর বলা হয়। কোনো ভেরিয়েবলের সাথে ইনক্রিমেন্ট ব্যবহার করলে তার মান ১ বেড়ে যায়। তবে ইনক্রিমেন্ট দুই ধরনের। প্রি-ইনক্রিমেন্ট এবং পোস্ট-ইনক্রিমেন্ট। যেমন– a=1; printf("%d",++a); printf("%d",a++); এখানে প্রথমে প্রি-ইনক্রিমেন্ট এবং পরে পোস্ট-ইনক্রিমেন্টের উদাহরণ দেয়া হয়েছে। প্রথমে a-এর মান বেড়ে ২ হবে, তারপর তা প্রিন্ট হবে। পরেরবার a-এর ভ্যালু প্রিন্ট হবে, তারপর তার ভ্যালু বেড়ে ৩ হবে। অর্থাৎ এই তিনটি স্টেটমেন্টের আউটপুট হবে ২২ (প্রথমে ২, তারপর আবার ২ প্রিন্ট করবে)। ডিক্রিমেন্ট অপারেটরের (--) ক্ষেত্রেও একই ধরনের নিয়ম প্রযোজ্য।

+= এই অপারেটরের নাম ইনক্রিমেন্ট অ্যান্ড অ্যাসাইন। a=3; a+=2; এই দুটি স্টেটমেন্টের মানে হলো প্রথমে a ভেরিয়েবলের মান ৩ নির্ধারণ হবে। তারপর a-এর মান ২ বেড়ে যাবে, অর্থাৎ ৫ হবে। দ্বিতীয় স্টেটমেন্টকে a=a+2; এভাবেও লেখা যায়। একইভাবে -=, /=, *= ইত্যাদি অপারেটরও ব্যবহার করা যায়। অন্যান্য অপারেটরের ক্ষেত্রেও একই নিয়ম প্রযোজ্য।

উপরে দেখানো হলো কিভাবে পয়েন্টার ভেরিয়েবল দিয়ে বিভিন্ন ভেরিয়েবল নিয়ন্ত্রণ করা যায়, পয়েন্টার দিয়ে কিভাবে এক্সপ্রেশন তৈরি করা যায় এবং পয়েন্টারের সাথে কিভাবে যোগ-বিয়োগ করা যায়। কোনো পয়েন্টার যদি একটি নির্দিষ্ট অ্যাড্রেসকে পয়েন্ট করে এবং সেই পয়েন্টারের সাথে যদি ৫ যোগ করা হয়, তাহলে ওই পয়েন্টারটি বর্তমান অ্যাড্রেস থেকে ৫ ঘর পরের অ্যাড্রেসকে পয়েন্ট করবে। এটিই পয়েন্টারের সাথে সরাসরি সংখ্যা যোগ-বিয়োগ করার মূল তত্ত্ব। কিন্তু পয়েন্টারের সাথে সরাসরি যোগ-বিয়োগের কাজ যতটা সহজে দেখানো হলো, আসলে ততটা সহজে হয় না। ধরা যাক, একটি পয়েন্টার বর্তমানে ৫৬৬১ অ্যাড্রেসকে পয়েন্ট করছে। তাহলে পয়েন্টারের সাথে যদি এখন ১ সরাসরি যোগ করা হয়, তাহলে স্বভাবতই তার ৫৬৬২ অ্যাড্রেসকে পয়েন্ট করার কথা। কিন্তু এটি সবসময় সত্য নাও হতে পারে। আসলে ব্যাপারটি নির্ভর করে পয়েন্টারের ডাটা টাইপের ওপর। যদি এটি একটি ক্যারেক্টার টাইপের পয়েন্টার হয়, তাহলে এটি ৫৬৬২-কে পয়েন্ট করবে। কিন্তু এটি ইন্টিজার টাইপের হলে ৫৬৬১+২ বা ৫৬৬৩-কে পয়েন্ট করবে। একইভাবে এটি ডাবল টাইপের পয়েন্টার হলে ৫৬৬১+৮ বা ৫৬৬৯-কে পয়েন্ট করবে। অর্থাৎ পয়েন্টারটি ডাবল টাইপের হলে তার সাথে ১ যোগ করা হলেও প্রকৃতপক্ষে ৮ যোগ হবে। কারণ, ডাবল টাইপ ভেরিয়েবলের জন্য মেমরিতে ৮ বাইট অ্যালোকেট করা হয়। সুতরাং সবশেষে বলা যায়, যেকোনো পয়েন্টারের সাথে কোনো মান সরাসরি যোগ করা হলে সেই মানের সমানসংখ্যক ভেরিয়েবলের জন্য যতটুকু অতিরিক্ত মেমরি দরকার, তা ওই পয়েন্টারের সাথে যোগ হয়ে যায়। এ কারণেই মূলত পয়েন্টারের ডাটা টাইপ সতর্কতার সাথে নিয়ন্ত্রণ করা উচিত। কেননা পয়েন্টার ডাটা টাইপই আসলে এ বিষয়টি নিয়ন্ত্রণ করে। একই নিয়ম অন্যান্য অপারেশনের ক্ষেত্রেও প্রযোজ্য।

পয়েন্টারের মাধ্যমে এমন কাজ করা যায়, যা অন্য কোনো কিছু দিয়ে করা সম্ভব নয়। পয়েন্টারের ব্যবহার একদিকে যেমন জটিল, অন্যদিকে তেমন গুরুত্বপূর্ণ। এ কারণেই পয়েন্টারকে অ্যাডভান্সড সি-এর অন্যতম বৈশিষ্ট্য বলা হয় কজ

ফিডব্যাক : wahid_cseaust@yahoo.com

ঘোষণা

কারুকাজ বিভাগের জন্য প্রোগ্রাম ও সফটওয়্যার টিপস বা টুকিটাকি লিখে পাঠান। লেখা এক কলামের মধ্যে হলে ভালো হয়। সফট কপিসহ প্রোগ্রামের সোর্স কোডের হার্ড কপি প্রতি মাসের ২০ তারিখের মধ্যে পাঠাতে হবে।