সহজ ভাষায় প্রোগ্রামিং সি/সি++

আহমদ ওয়াহিদ মাসুদ

সি প্রোগ্রামিং ল্যাঙ্গুয়েজে ফাইল নিয়ে কাজ করার জন্য অনেক ধরনের সুবিধা রয়েছে। ফাইল বা ডাটা যেনো নষ্ট হয়ে না যায় সেজন্যও ব্যবস্থা রয়েছে, যা আগের লেখায় তুলে ধরা হয়েছে। এ লেখায় বাফার ফ্ল্যাশ, কমান্ড লাইন ও প্রি-প্রসেসর ডিরেক্টিভ নিয়ে আলোচনা করা হয়েছে।

ডিস্ক বাফার ফ্ল্যাশ করা : প্রোগ্রামিংয়ে বাফার অনেক গুরুত্বপূর্ণ একটি বিষয়। যেকোনো প্রোগ্রাম, যেগুলো সাধারণত ফাইল নিয়ে কাজ করে, সেগুলোকে চলার জন্য বাফার ব্যবহার করতে হয়। সি-তে বাফারের কাজ জানার আগে বাফার আসলে কী সে সম্পর্কে একটু ধারণা থাকা দরকার। বাফার হলো মেমরির কিছু অংশ, যা অস্থায়ীভাবে ডাটা রাখার জন্য ব্যবহার হয়। বাফারের সাইজ সাধারণত ডিস্ক সেক্টরের সমান হয়ে থাকে। ফাইলে কোনো ডাটা লেখার জন্য সি-তে যেসব লাইব্রেরি ফাংশন ব্যবহার হয়, বাফার পূর্ণ হলে বা ফাইল বন্ধ করা হলে সেসব লাইব্রেরি ফাংশন ডাটাগুলো ফাইলে লিখে ফেলে। মূলত বারবার ডিস্কে যেনো I/O অপারেশন করতে না হয়, সেজন্য এ পদ্ধতি ব্যবহার করা হয়। তবে এই বাফার পদ্ধতি ব্যবহার করলে ডাটা ডিস্কে ঠিকমতো নাও লেখা হতে পারে। যেমন, fputs() ফাংশনের মাধ্যমে কোনো ডাটা লেখা হলে এবং ফাংশনটি যদি ইওএফ রিটার্ন না করে, তাহলে ধরে নেয়া হয় ডাটাগুলো ঠিকমতো ফাইলে লেখা হয়েছে। কিন্তু সবসময় এটা ঠিক নাও হতে পারে। ডাটা হয়তো মেমরিতেই থেকে গেল এবং এ সময় যদি কোনো কারণে কমপিউটার বন্ধ হয়ে যায়, তাহলে সব ডাটা নষ্ট হয়ে যাবে। এ ক্ষেত্রে ডাটা আসলেই ফাইলে লেখা হয়েছে কি না, তা নিশ্চিত হওয়ার জন্য বাফার ফ্ল্যাশ করতে হয়। এ কাজটির জন্য fflush() ফাংশন ব্যবহার করা যায়। ফাংশনের প্যারামিটার হিসেবে শুধু নির্দিষ্ট ফাইল পয়েন্টার ব্যবহার করলেই হবে। যেমন :

```
int main{
fputs(str,fp)
fflush(fp);}
```

এখানে প্রথমে একটি স্ট্রিং ভেরিয়েবল এফপি পয়েন্টারের মাধ্যমে লেখা হচ্ছে এবং পরে এফফ্ল্যাশের মাধ্যমে ওই পয়েন্টারের বাফার ফ্ল্যাশ করে দেয়া হচ্ছে। এফফ্ল্যাশ ফাংশন যদি এফপুটএস বা এ ধরনের কোনো ফাংশনের পরে ব্যবহার করা হয়, তাহলে সাথে সাথে ডাটা ফাইলে লেখা হয়। এফফ্ল্যাশ ঠিকমতো কাজ করলে ০ রিটার্ন করবে, অন্যথায় ইওএফ রিটার্ন করবে।

ফাংশন প্যারামিটার হিসেবে ফাইল পয়েন্টার

সি-তে অন্য যেকোনো সাধারণ পয়েন্টার ভেরিয়েবলের মতো ফাইল পয়েন্টারকেও ফাংশনের প্যারামিটার হিসেবে পাঠানো যায়। যেমন :

```
void functionx(FILE* fp){
……}
FILE* fpln;
fpln=fopen(“test.txt”,”r”);
function(fpln);
```

এখানে প্রথমে ফাংশনএক্স নামে একটি ফাংশন খোলা হয়েছে, যার প্যারামিটার প্রটোটাইপ হিসেবে একটি ফাইল পয়েন্টার দেয়া হয়েছে। পরে অপর একটি ফাইল পয়েন্টার খুলে তাকে ওই ফাংশনের প্যারামিটার হিসেবে পাঠানো হয়েছে।

কমান্ড লাইন আর্গুমেন্ট

কোনো প্রোগ্রাম চালাতে হলে ওই প্রোগ্রামের .exe/.comবিশিষ্ট ফাইলের নাম ডসের কমান্ড প্রম্পটে লিখতে হয়। যেমন, টার্বো সি চালাতে হলে TC ডিরেক্টরিতে ঢুকে এর .exe ফাইল চালাতে হয়। যেমন :

কমান্ড প্রম্পট ওপেন করে,

```
C:\TC\BIN\tc
```

এটি লিখে এন্টার চাপলে টার্বো সি চালু হয়ে যাবে। আর কমান্ড প্রম্পট আনতে হলে স্টার্ট মেনুর সার্চে গিয়ে cmd লিখলেই হবে। কিন্তু কিছু কিছু প্রোগ্রাম আছে, যেগুলোর ইএক্সই ফাইল চালাতে হলে ফাইলের নামের পাশে এক বা একাধিক প্যারামিটার বা আর্গুমেন্ট লিখতে হয়। উদাহরণ হিসেবে ডসের কপি করার প্রোগ্রামের কথা বলা যেতে পারে। এ প্রোগ্রামটি চালাতে দুটি আর্গুমেন্ট লিখতে হয়। সোর্সের লোকেশন হলো প্রথম আর্গুমেন্ট, আর ডেস্টিনেশন হলো দ্বিতীয় আর্গুমেন্ট। যেমন :

```
C:\> copy c:\autoexe.bat d:\
```

এখানে দেখা যাচ্ছে প্রথমে কপি কমান্ড দেয়া হলো। এরপর সোর্স ফাইল হিসেবে সি ড্রাইভের একটি ফাইল দেয়া হলো এবং ডেসটিনেশন হিসেবে ডি ড্রাইভ দেয়া হলো। এভাবে কমান্ড প্রম্পটে আর্গুমেন্টসহ কোনো কমান্ড দেয়াকে কমান্ড লাইন আর্গুমেন্ট বলে। এখন দেখানো হবে সি-তে কীভাবে এ ধরনের প্রোগ্রাম লেখা যায়, যাতে প্রোগ্রামও প্রয়োজনানুসারে এক বা একাধিক আর্গুমেন্ট নিতে পারে।

আমরা জানি, প্রতিটি সি প্রোগ্রামে মেইন নামে একটি ফাংশন থাকে। এ পর্যন্ত যতগুলো মেইন ফাংশন দেখানো হয়েছে, এর সবগুলোতেই মেইন ফাংশনের প্যারামিটার খালি রাখা হয়েছে বা আগে ভয়েড লেখা হয়েছে। তবে মেইন ফাংশনের প্যারামিটার হিসেবে একটি ইন্টিজার ও একটি ক্যারেক্টার অ্যারে ব্যবহার করা যায়, যাদের মান কমান্ড লাইন থেকে নেয়া হবে। যেমন :

```
Int main(int argc, char* argv[]){
….}
```

প্রোগ্রামের exe-এর পরে কয়টি আর্গুমেন্ট লেখা হয়েছে তা argc-এর মাধ্যমে এবং যে আর্গুমেন্ট লেখা হয়েছে তা argv[]-এর মাধ্যমে পাওয়া যায়। যেমন : কমান্ড প্রম্পটে যদি লেখা হয়,

```
C:\ copytext temp.dat temp.sav
```

তাহলে এখানে argc-এর মান হবে ৩ ও argv-এর এলিমেন্ট সংখ্যা হবে ৩। এ ক্ষেত্রে প্রতিটি এলিমেন্টের মান হবে বিভিন্ন আর্গুমেন্টগুলো।

প্রি-প্রসেসর ডিরেক্টিভ

সি ল্যাঙ্গুয়েজের কিছু অনন্য বৈশিষ্ট্য রয়েছে, যার মাধ্যমে একজন প্রোগ্রামার অনেক সহজে ও অনেক তাড়াতাড়ি প্রোগ্রাম লিখতে পারেন। অনেক ল্যাঙ্গুয়েজেই এ ধরনের বৈশিষ্ট্য দেখা যায়। এখানে সে ধরনেরই একটি বৈশিষ্ট্য প্রসেসর ডিরেক্টিভ সম্পর্কে আলোচনা করা হয়েছে।

সহজভাবে বলা যায়, প্রোগ্রামের কোথাও যদি # ক্যারেক্টারের পর কোনো কিছু লেখা হয়, তাহলে তাকেই প্রসেসর ডিরেক্টিভ বলে। কোনো প্রোগ্রামকে যখন কম্পাইল করা হয়, তখন কম্পাইলার ওই প্রোগ্রামকে প্রসেস করার আগে প্রি-প্রসেসর নামে অন্য একটি সফটওয়্যার ওই প্রোগ্রামকে প্রসেস করে। এই সফটওয়্যারের কাজ হলো প্রোগ্রামে লেখা বিভিন্ন হেডার ফাইলকে সংযুক্ত করা বা কোনো কনস্ট্যান্টকে মূল ভ্যালু দিয়ে প্রতিস্থাপন করা। যেহেতু কম্পাইল করার আগেই এ সফটওয়্যারটি প্রোগ্রামকে প্রসেস করে, তাই এর নাম দেয়া হয়েছে প্রি-প্রসেসর। একটি ছোট উদাহরণ নিচে দেয়া হলো :

```
#include <stdio.h>
#define value 128
int main()
{
….
….
}
```

এই প্রোগ্রামটি যখন কম্পাইল করা হবে, তখন প্রি-প্রসেসর সফটওয়্যারটি এই প্রোগ্রামের মেইন ফাংশনে লেখা প্রতিটি ভ্যালু নামের ভেরিয়েবলকে ১২৮ দিয়ে প্রতিস্থাপন করবে। এছাড়া প্রোগ্রামের কোডের শুরুতে stdio.h ফাইলের কোডগুলোকেও সংযোজন করে দেবে। তাই বলা হয়, প্রি-প্রসেসর প্রোগ্রামের #include, #define mn # ক্যারেক্টারের যেসব স্টেটমেন্ট নিয়ে কাজ করে তাদেরকে প্রি-প্রসেসর ডিরেক্টিভ বলা হয়।

উপরে সংজ্ঞা দেয়ার সময় বলা হয়েছে, যেসব লাইনের শুরুতেই # থাকে, এর মানে কিন্তু এই নয়, ইউজার তার ইচ্ছেমতো যেকোনো লাইনের শুরুতে # ক্যারেক্টার ব্যবহার করতে পারেন। সি-তে অন্য সব উপাদানের মতো প্রি-প্রসেসর ডিরেক্টিভ ব্যবহার করারও কিছু নির্দিষ্ট নিয়ম রয়েছে। প্রি-প্রসেসর ডিরেক্টিভ সবসময় # ক্যারেক্টার দিয়ে শুরু হবে এবং এদের শেষে সেমিকোলন দেয়া যাবে না। অ্যান্সি স্ট্যান্ডার্ড অনুযায়ী # ক্যারেক্টারটি যেকোনো কলামে হতে পারবে। নিচে বিভিন্ন প্রি-প্রসেসর ডিরেক্টিভ নিয়ে আলোচনা করা হলো।

(বাকি অংশ ৬৪ পৃষ্ঠায়)