পঞ্চম অধ্যায়

প্রোগ্রামিং ভাষা

Programming Language

প্রোগ্রামিং প্রতিযোগিতায় অংশগ্রহণকারী শিক্ষার্থীরা

কম্পিউটার নামক যন্ত্রটি কোনো না কোনোভাবে পুরো পৃথিবীর প্রায় প্রতিটি মানুষের জীবনকে প্রভাবিত করেছে। এই অসাধারণ যন্ত্রটি কোন কাজে ব্যবহার করা যাবে সেটি শুধু মানুষের সৃজনশীলতা দিয়ে সীমাবদ্ধ। তবে এককভাবে কম্পিউটার নামের এই যন্ত্রটির সাথে জন্য আরেকটি যন্ত্রের কোনো পার্থক্য নেই। কম্পিউটার আলাদাভাবে একটি বিশেষ কিছু হয়ে উঠে কারণ এটিকে নির্দিষ্ট কোনো কাজ করার জন্য প্রোগ্রাম করা সম্ভব। কম্পিউটার যেহেতু একটি ইলেকট্রনিক যন্ত্র ছাড়া আর কিছুই নয় এবং সেটি 1 এবং 0 ছাড়া আর কিছুই বুঝতে পারে না, তাই তাকে প্রোগ্রাম করার জন্য এই 1 এবং 0 দিয়েই মেশিন কোডে কিছু দুর্বোধ্য নির্দেশনা দিতে হয়। বিষয়টিকে সহজ করার জন্য অনেক প্রোগ্রামিং ভাষা উদ্ভাবন করা হয়েছে, এই ভাষাগুলোতে একজন প্রয়োজনীয় কোড লিখতে পারে যেটি পরবর্তীকালে মেশিন কোডে রূপান্তরিত করে কম্পিউটারের কাছে নির্দেশনা হিসেবে পাঠানো হয়। এরকম একটি জনপ্রিয় এবং বহুল ব্যবহৃত কম্পিউটার প্রোগ্রামিংয়ের ভাষা হচ্ছে সি (C)। এই অধ্যায়ে শিক্ষার্থীদের কাছে প্রোগ্রামিংয়ের খুঁটনাটির সাথে সাথে C ভাষায় প্রোগ্রামিং করার প্রাথমিক বিষয়গুলো তুলে ধরা হয়েছে।

এ অধ্যায় পাঠ শেষে শিক্ষার্থীরা—

- প্রোগ্রামের ধারণা ব্যাখ্যা করতে পারবে;
- বিভিন্ন স্তরের প্রোগ্রামিং ভাষা বর্ণনা করতে পারবে।

ব্যাবহারিক

- প্রোগ্রামের সংগঠন প্রদর্শন করতে পারবে;
- প্রোগ্রাম অ্যালগরিদম ও ফ্লো চার্ট প্রস্তুত করতে পারবে;
- 'সি' প্রোগ্রামিং ভাষা ব্যবহার করে প্রোগ্রাম প্রস্তুত করতে পারবে।

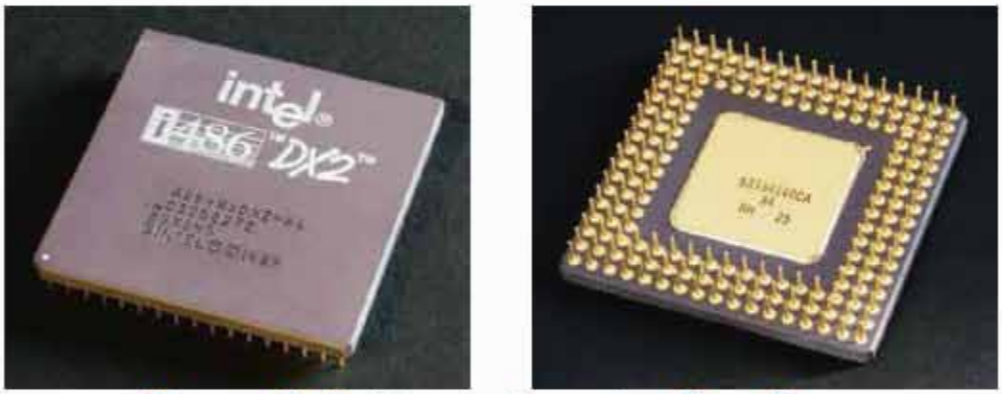
৫.১ প্রোগ্রামের ধারণা (Concept of Programming)

কম্পিউটারে একটি কেন্দ্রীয় প্রক্রিয়াকরণ অংশ বা সেন্ট্রাল প্রসেসিং ইউনিট (Central Processing Unit), সংক্ষেপে সিপিইউ (CPU) রয়েছে। বর্তমানে আমরা যেটি মাইক্রোপ্রসেসর হিসেবে চিনি তারমধ্যে এক বা একাধিক সিপিইউ থাকে। এই সিপিইউ-এর কাজ হচ্ছে বিভিন্ন হিসাব-নিকাশ করা।

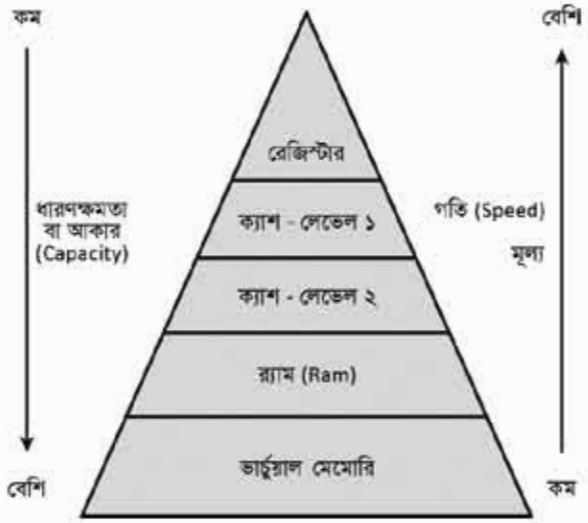
চিত্র 5.1 : কম্পিউটারের গঠন

মাইক্রোপ্রসেসরে যেসব হিসাব-নিকাশ করা হয় তারমধ্যে রয়েছে যোগ-বিয়োগ-গুণ-ভাগসহ বিভিন্ন অপারেশন। এই অপারেশনগুলো যেসব ডেটার উপর করা হয় সেসব ডেটা সংরক্ষণ করার জন্য প্রয়োজন হয় কম্পিউটার মেমোরির। আবার অপারেশন শেষে অপারেশনের ফলাফলও এই কম্পিউটার মেমোরিতে সংরক্ষণ করা হয়।

কম্পিউটারের মেমোরিকে সাধারণত দুই ভাগে ভাগ করা যায় : অস্থায়ী ও স্থায়ী। ইংরেজিতে বলে ভোলাটাইল (volatile) ও নন-ভোলাটাইল (non-volatile)। যেসব মেমোরিতে কম্পিউটার বন্ধ করার পরেও ডেটা সংরক্ষিত থাকে, তাকে বলে স্থায়ী (non-volatile) মেমোরি। যেমন : হার্ডডিস্ক, রম, ডিভিডি, ইউএসবি ড্রাইভ ইত্যাদি। আর যেসব মেমোরির ডেটা কম্পিউটার বন্ধ (ক্ষেত্রবিশেষে প্রোগ্রাম বন্ধ) করলে হারিয়ে যায়, সেগুলোকে বলে অস্থায়ী মেমোরি। যেমন : র‍্যাম (RAM)। কম্পিউটার প্রোগ্রামগুলো ডেটা নিয়ে কাজ করার সময় অস্থায়ী মেমোরি ব্যবহার করে। স্থায়ী মেমোরিগুলো বেশ ধীরগতির হয় বলে সেগুলো ব্যবহার করা হয় না।



চিত্র 5.2 : একটি মাইক্রোপ্রসেসরের পেছনের ও সামনের দিকের ছবি



চিত্র 5.3 : কম্পিউটারের অস্থায়ী মেমোরির বিভিন্ন পর্যায়

কম্পিউটারের প্রসেসরের মধ্যেও কিছু মেমোরি আছে, প্রসেসরের সবচেয়ে কাছে থাকে রেজিস্টার, আর তার পরেই থাকে ক্যাশ মেমোরি। রেজিস্টারের চেয়ে ক্যাশ মেমোরির আকার বড়, মানে বেশি তথ্য ধারণ করতে পারে, তবে গতি একটু কম। রেজিস্টার ও ক্যাশ মেমোরি প্রসেসরের মধ্যেই যুক্ত করা থাকে। তারপরে আসে র‍্যাম। র‍্যাম প্রসেসরের বাইরে মাদারবোর্ডে সংযুক্ত থাকে। ক্যাশের তুলনায় র‍্যামের আকার বেশ বড়, তবে গতি কম।

খরচের দিক থেকে সবচেয়ে ব্যয়বহুল রেজিস্টার মেমোরি, তারপরে ক্যাশ মেমোরি। র‍্যাম তাদের তুলনায় বেশ সস্তা। র‍্যামের পরে আসে ভার্চুয়াল মেমোরি। র‍্যামে যখন জায়গা হয় না, তখন হার্ডডিস্কের একটা অংশকে কম্পিউটারের অপারেটিং সিস্টেম মেমোরি হিসেবে ব্যবহার করতে দেয়। সেটি অবশ্যই র‍্যামের তুলনায় অনেক ধীর গতির।

৫.২ প্রোগ্রামের ভাষা (Language of Programming)

কম্পিউটারকে দিয়ে কোনো কাজ করাতে হলে তাকে বিশেষভাবে নির্দেশ দিতে হয়। কম্পিউটারের প্রসেসর কেবল একটি নির্দিষ্ট সেটের কমান্ড এক্সিকিউট করতে পারে, যাকে বলে ইনস্ট্রাকশন সেট। কিন্তু প্রোগ্রামাররা সাধারণত সেই ভাষায় প্রোগ্রাম লেখেন না, বরং প্রোগ্রাম তৈরি করার জন্য শত শত প্রোগ্রামিং ভাষা চালু আছে।

বিভিন্ন দশকে উদ্ভাবিত কিছু গুরুত্বপূর্ণ প্রোগ্রামিং ভাষা

প্রোগ্রামিং ভাষার নাম	আবিষ্কারের সাল
ফোরট্রান (Fortran)	1954-57
লিসপ (Lisp)	1956-59
কোবোল (Cobol)	1959-60
বেসিক (Basic)	1964
প্যাসকেল (Pascal)	1970
সি (C)	1972
সি++ (C++)	1983
পার্ল (Perl)	1987
পাইথন (Python)	1989

প্রোগ্রামিং ভাষার নাম	আবিষ্কারের সাল
ভিজুয়াল বেসিক (Visual Basic)	1991
পিএইচপি (PHP)	1995
জাভা (Java)	1995
জাভাস্ক্রিপ্ট (Javascript)	1995
স্কালা (Scala)	2003
গো (Go)	2009
রাস্ট (Rust)	2010
কটলিন (Kotlin)	2011

৫.২.১ মেশিন ভাষা (Machine Language)

কম্পিউটারের প্রসেসর বাইনারি সংখ্যা পদ্ধতি ব্যবহার করে বিভিন্ন হিসেব করে। বাইনারি সংখ্যা পদ্ধতিতে কেবল দুটি অঙ্ক রয়েছে— 1 ও 0। এই দুটি অঙ্ক ব্যবহার করেই প্রসেসরের জন্য বিশেষ সংকেত তৈরি করা হয়। 0 ও 1 দিয়ে তৈরি যে প্রোগ্রাম, তাকে বলে মেশিন কোড (machine code), আর এই ভাষাটিকে বলা হয় মেশিন ল্যাঙ্গুয়েজ। কম্পিউটারের জন্য মেশিন কোড খুব সহজবোধ্য হলেও মানুষের জন্য মেশিন ল্যাঙ্গুয়েজের কোড পড়া দুঃসাধ্য

চিত্র 5.4: পৃথিবীর প্রথম প্রোগ্রামার অ্যাডা লাভলেস (1815-1852)

ব্যাপার। কারণ কোডে কেবল 0 আর 1 থাকে। তাই মানুষের পক্ষে এই ভাষায় বড় প্রোগ্রাম তৈরি করা অসম্ভব বলা চলে।

কম্পিউটার প্রোগ্রামিংকে সহজ করার জন্য বিভিন্ন প্রসেসর নির্মাতা প্রতিষ্ঠান তাদের প্রসেসরের সঙ্গে তৈরি করেন একটি ইনস্ট্রাকশন সেট। ইনস্ট্রাকশন সেটে কিছু সহজ ইনস্ট্রাকশন দিয়ে দেওয়া হলো যেগুলো ব্যবহার করে প্রসেসরকে নির্দেশ দেওয়া যায় বা প্রোগ্রাম তৈরি করা যায়। কেবল 0 আর 1 ব্যবহার করার চেয়ে ইনস্ট্রাকশন সেট ব্যবহার করে প্রোগ্রাম লেখা অপেক্ষাকৃত সহজ হয়।

৫.২.২ অ্যাসেম্বলি ভাষা (Assembly Language)

প্রোগ্রামারদের জন্য প্রোগ্রাম লেখা সহজতর করার জন্য মেশিন ল্যাঙ্গুয়েজের পর তৈরি হলো অ্যাসেম্বলি ল্যাঙ্গুয়েজ। এটি একটি প্রোগ্রামিং ভাষা। মেশিন ল্যাঙ্গুয়েজের চেয়ে এই ভাষায় প্রোগ্রাম লেখা ও পড়া প্রোগ্রামারদের জন্য সহজ। কম্পিউটারের প্রসেসর কিন্তু সরাসরি অ্যাসেম্বলি ল্যাঙ্গুয়েজ দিয়ে তৈরি প্রোগ্রাম রান করতে পারে না। অ্যাসেম্বলি ল্যাঙ্গুয়েজে লেখা কোডকে আগে মেশিন কোডে রূপান্তর করতে হয়, তারপর প্রসেসর সেটিকে এক্সিকিউট করতে পারে। অ্যাসেম্বলি ল্যাঙ্গুয়েজে লেখা কোডকে মেশিন কোডে রূপান্তর করার কাজটি করে যে প্রোগ্রাম, তার নাম অ্যাসেম্বলার (assembler)।

৫.২.৩ মধ্যম স্তরের ভাষা (Mid-Level Language)

অ্যাসেম্বলি ল্যাঙ্গুয়েজ এবং উচ্চস্তরের ভাষার মধ্যবর্তী ভাষাকে মধ্যম স্তরের ভাষা বলে। এটি কম্পিউটারের হার্ডওয়্যার এবং প্রোগ্রামিংয়ের মাঝে একটি সেতু বন্ধন তৈরি করে দেয়। সি ল্যাঙ্গুয়েজ মধ্যম স্তরের ভাষার একটি চমৎকার উদাহরণ কারণ এটি দিয়ে একদিকে অপারেটিং সিস্টেমের মতো সিস্টেম প্রোগ্রামিং করা যায় অন্যদিকে তেমনি দৈনন্দিন ব্যবহারের জন্য অ্যাপ্লিকেশন সফটওয়্যার তৈরি করা যায়।

৫.২.৪ উচ্চ স্তরের ভাষা (High Level Language)

মেশিন ল্যাঙ্গুয়েজ ও অ্যাসেম্বলি ল্যাঙ্গুয়েজ হচ্ছে লো-লেভেল প্রোগ্রামিং ভাষা। অ্যাসেম্বলি ল্যাঙ্গুয়েজ প্রোগ্রামারদের জন্য আগের চেয়ে সহজে প্রোগ্রাম লেখার ব্যবস্থা করলেও এ ভাষায় বড় বড় প্রোগ্রাম লেখাটা অনেক কঠিন এবং সময়-সাপেক্ষ। প্রোগ্রামিং ব্যবহার করে মানুষ যখন বিভিন্ন ধরনের সমস্যার সমাধান করতে লাগল, তখন প্রয়োজন হলো এমন ধরনের প্রোগ্রামিং ভাষার, যে সব ভাষায় প্রোগ্রাম লেখা ও পড়া মানুষের জন্য অনেক বেশি সহজ হবে। তখন তৈরি হলো উচ্চ স্তরের প্রোগ্রামিং ভাষা। কোবল (Cobol), ফোরট্রান (Fortran), সি (C) ইত্যাদি প্রোগ্রামিং ভাষার আবিষ্কারের ফলে প্রোগ্রামিং ভাষা অনেকখানি বদলে গেল। এসব ভাষা ব্যবহার করে বিভিন্ন সমস্যা আগের চেয়ে অনেক দ্রুত প্রোগ্রাম লিখে সমাধান করা যেত। তাই এসব ভাষাকে উচ্চস্তরের প্রোগ্রামিং ভাষা বলা হতো। তবে সময়ের সঙ্গে আরো নতুন নতুন প্রোগ্রামিং ভাষা তৈরি হলো, যেগুলো প্রোগ্রামিং ভাষাকে আরো সহজবোধ্য করল এবং এসব ভাষা ব্যবহার করে প্রোগ্রাম ডিজাইন করাও সহজ হলো। যেমন— সি প্লাস প্লাস (C++), জাভা (Java), সি শার্প (C#), পিএইচপি (PHP), পাইথন (Python) ইত্যাদি। বর্তমানে এগুলোকে হাই লেভেল প্রোগ্রামিং ভাষা হিসেবে বিবেচনা করা হয়।

সি (C) : সি একটি সাধারণভাবে ব্যবহারের উপযোগী অত্যন্ত জনপ্রিয় প্রোগ্রামিংয়ের ভাষা। 1972 সালে ডেনিস রিচি (Dennis Ritchie) বেল ল্যাবে এই ভাষাটি তৈরি করেন। বলা হয়ে থাকে এই ভাষাটি জানা থাকলে কম্পিউটারের অন্য যে কোনো ভাষা শেখা খুব সহজ। সি ল্যাঙ্গুয়েজ দিয়ে অপারেটিং সিস্টেম থেকে জটিল ডাটাবেজ ম্যানেজমেন্ট প্রোগ্রাম, ইন্টারনেট ব্রাউজার কিংবা ইন্টারপ্রেটার পর্যন্ত সবকিছু তৈরি করা যায়। এটি একটি চমৎকার স্ট্রাকচার্ড প্রোগ্রামিং ভাষা, এখানে ছোট ছোট অসংখ্য অংশকে সমন্বয় করে একটি জটিল প্রোগ্রাম তৈরি করা যায়।

সি প্লাস প্লাস (C++) : প্রোগ্রামিংয়ের জগতে ক্লাস একটি গুরুত্বপূর্ণ ধারণা। একই ধরনের বৈশিষ্ট্য রয়েছে সেরকম কিছুকে ক্লাস বলে অভিহিত করা হয়। সি প্রোগ্রামিং ল্যাঙ্গুয়েজের সাথে ক্লাস সংযুক্ত করে এবং পরে আরো নতুন কিছু বৈশিষ্ট্য যোগ করে C++ ল্যাঙ্গুয়েজের সূচনা হয়। 1980 সালে বেল ল্যাবে কর্মরত জর্ন স্ট্রাউস্ট্রপ (Bjarne Stroustrop) এই ভাষাটি উদ্ভাবন করেন। একজন প্রোগ্রামারকে পুরোপুরি নিজের মতো প্রোগ্রামিং করার স্বাধীনতা দেওয়া এই ভাষাটির একটি মূল নীতি।

ভিজুয়াল বেসিক (Visual Basic) : 1991 সালে মাইক্রোসফট তাদের উইন্ডোজ অপারেটিং সিস্টেমে প্রোগ্রামিং করার জন্য ভিজুয়াল বেসিক ল্যাঙ্গুয়েজ উদ্ভাবন করেছিল। এটি মোটামুটি একটি স্থিতিশীল পর্যায়ে পৌঁছানোর সাথে সাথে ব্যাপক জনপ্রিয়তা লাভ করেছিল। অত্যন্ত সহজে এর প্রোগ্রামিং করার কারণে এবং প্রোগ্রামের পরিবর্তন করা হলে পুনরায় কম্পাইল না করেই প্রোগ্রাম চালানোর সুবিধার জন্য প্রোগ্রামার এবং সাধারণ ব্যবহারকারী সবার কাছে সমান জনপ্রিয় ছিল।

জাভা (Java) : 1991 সালে সান মাইক্রো সিস্টেম জাভা প্রোগ্রামিং ভাষার সূচনা করে। এটি বর্তমানে একটি জনপ্রিয় ভাষা। এর একটি প্রধান বৈশিষ্ট্য হচ্ছে এটি একটি প্লাটফর্মে কম্পাইল করে নিলে জাভা ব্যবহার করে সেরকম অন্য যে কোনো প্লাটফর্মে সরাসরি ব্যবহার করা যায় (WORA: Write Once, Run Anywhere)। গুরুত্বপূর্ণ ওয়েব ব্রাউজারগুলো ওয়েব পেজের ভেতর জাভা অ্যাপলেট চালু করার সক্ষমতা দেওয়ার কারণে এটি খুবই দ্রুত সবার কাছে জনপ্রিয় হয়ে উঠে।

অ্যালগল (Algol) : ইউরোপ এবং আমেরিকার বেশ কিছু কম্পিউটার বিজ্ঞানীদের সম্মিলিত প্রচেষ্টায় 1958 সালে ALGOL (Algorithmic Language) প্রোগ্রামিং ভাষাটি জন্ম নেয়। সেই সময়ের অন্য প্রোগ্রামিং ভাষার তুলনায় এটি অনেক বেশি ভবিষ্যৎমুখী এবং আধুনিক একটি প্রোগ্রামিং ভাষা ছিল। এমনকি বর্তমানের আধুনিক প্রোগ্রামিং ভাষার সিন্টেক্সেও অ্যালগল ভাষার ছাপ লক্ষ করা যায়। বিজ্ঞান এবং গবেষণাতে অ্যালগল ব্যাপকভাবে ব্যবহার হলেও সহজ ইনপুট এবং আউটপুট প্রযুক্তির অভাবে ব্যবসা-বাণিজ্যের জগতে এটি তেমন সুপরিচিত হওয়ার সুযোগ পায়নি।

ফোরট্রান (Fortran) : 1957 সালে আইবিএম কোম্পানি বিজ্ঞান এবং প্রযুক্তির ক্ষেত্রে ব্যবহারের জন্য ফোরট্রান (Formula Translation) নামে একটি উচ্চস্তরের ভাষা উদ্ভাবন করে। এটি গাণিতিক বিশ্লেষণ করার জন্য বিশেষ পারদর্শী ছিল বলে বিজ্ঞানী এবং গবেষকরা এই ভাষাটিকে সাদরে গ্রহণ করে নেয়। একসময় পৃথিবীর প্রায় সব বৈজ্ঞানিক গবেষণায় এই ভাষা এককভাবে ব্যবহার কর হতো। শুনে অবিশ্বাস্য মনে হতে পারে কিন্তু অত্যন্ত দ্রুত হিসাব করতে পারে বলে বড় বড় সিমুলেশনে ব্যবহার করার জন্য এখনো এই ভাষাটি টিকে আছে। (দ্রুততায় এর কাছাকাছি অন্য ভাষাটি হচ্ছে C++) 2018 সালে ফোরট্রানের সর্বশেষ

ভার্সনটি রিলিজ করা হয়েছে। ফোরট্রান ব্যবহার করে পদার্থবিজ্ঞান এবং রসায়নের অনেক বড় বড় গাণিতিক সমস্যার সমাধান করে রাখায় এখনো তার কোনো কোনোটি বিজ্ঞানীরা তাদের গবেষণার কাজে ব্যবহার করেন।

পাইথন (Python) : গিডো ভান রসাম (Gido van Rossum) 1991 সালে পাইথন উদ্ভাবন করেন। এটি বর্তমানে সবচেয়ে জনপ্রিয় ভাষাগুলোর একটি এবং 2018 সালে এটি IEEE কর্তৃক সর্বশ্রেষ্ঠ প্রোগ্রামিং ভাষা হিসেবে স্বীকৃতি পেয়েছে। পাইথনের বৈশিষ্ট্য হচ্ছে এর অত্যন্ত সহজ এবং পাঠযোগ্য সিনট্যাক্স। এটি বিভিন্ন প্ল্যাটফর্মে চলে এবং ক্লাউডভিত্তিক ওয়েব অ্যাপ্লিকেশন, ডেটা অ্যানালাইসিস ও মেশিন লার্নিং অ্যাপ্লিকেশন তৈরিতে ব্যবহার করা হয়।

৫.২.৫ চতুর্থ প্রজন্মের ভাষা (4th Generation Language— 4GL)

প্রোগ্রামিংকে মানুষের জন্য আরো সহজ করার প্রচেষ্টা অব্যাহত থাকে এবং যার ফলে এমন প্রোগ্রামিং ভাষা তৈরি হয়, যেগুলো মানুষের ভাষার কিছুটা কাছাকাছি। এসব প্রোগ্রামিং ভাষাকে বলা হয় চতুর্থ প্রজন্মের ভাষা বা 4GL। ডেটাবেজ অধ্যায়ে যে SQL ভাষা দেখানো হয়েছে, সেটি হচ্ছে 4GL ভাষা। এ ছাড়াও যখন নানা ধরনের সফটওয়্যার টুলে গ্রাফিকেল ইন্টারফেস ব্যবহার করা হয়, একটি মেনু কিংবা বাটনে চাপ দিয়ে কিছু করে ফেলা যায়, তার পিছনেও চতুর্থ প্রজন্মের ভাষার অবদান আছে বলে বিবেচনা করা হয়।

৫.৩ অনুবাদক প্রোগ্রাম (Translator Program)

বর্তমানে হাজার খানেক প্রোগ্রামিং ভাষা প্রচলিত। যদিও সব ভাষা সমানভাবে জনপ্রিয় নয়। ভাষা যে রকমই হোক না কেন, কম্পিউটারের প্রসেসর 1 আর 0 ছাড়া কিছু বোঝে না। তাই বিভিন্ন ভাষায় লেখা প্রোগ্রামকে মেশিন কোডে রূপান্তর করতে হয়। এই কাজটি করার জন্য বিশেষ প্রোগ্রাম তৈরি করা হয়, যাকে বলে অনুবাদক প্রোগ্রাম। নিচে তিন ধরনের অনুবাদকের কথা বলা হলো:

অ্যাসেম্বলার (Assembler) : অ্যাসেম্বলি ভাষায় লেখা প্রোগ্রামকে মেশিন কোডে অনুবাদ করে অ্যাসেম্বলার নামক একটি প্রোগ্রাম।

উচ্চ স্তরের যেসব প্রোগ্রামিং ভাষা, সেগুলোকে মেশিন কোডে অনুবাদ করার কাজটি করার জন্য দু ধরনের প্রোগ্রাম রয়েছে— কম্পাইলার (Compiler) ও ইন্টারপ্রেটার (Interpreter)। প্রতিটি উচ্চ স্তরের প্রোগ্রামিং ভাষারই পৃথক কম্পাইলার অথবা ইন্টারপ্রেটার রয়েছে। এই দুই ধরনের অনুবাদক প্রোগ্রামের উদ্দেশ্য এক হলেও কাজের ধরনে কিছুটা ভিন্নতা রয়েছে।

কম্পাইলার (Compiler) : কম্পাইলার প্রথমে পুরো প্রোগ্রামটি পরীক্ষা করে দেখে যে এর ভাষার নিয়মকানুন (যাকে ইংরেজিতে বলে সিনট্যাক্স Syntax) ঠিক আছে কি না। যদি ঠিক থাকে, তখন সে পুরো প্রোগ্রামটি কম্পাইল করে মেশিন কোডে রূপান্তর করে। যেহেতু পুরো প্রোগামটি একবারে কম্পাইল করা হয় তাই প্রোগ্রামে কোনো ভুল থাকলে সব একসাথে দেখানো হয়। সে কারণে ভুলগুলো শুদ্ধ করা একটু জটিল। তবে কম্পাইল করার পর এই প্রোগ্রামগুলো অনেক দ্রুতগতিতে কাজ করে।

ইন্টারপ্রেটার (Interpreter) : ইন্টারপ্রেটার পুরো প্রোগ্রাম পরীক্ষা না করে প্রোগ্রামের প্রতিটি স্টেটমেন্ট (statement বা নির্দেশ) মেশিন কোডে রূপান্তর করে সেটিকে এক্সিকিউট করে। অর্থাৎ কোনো প্রোগ্রামে যদি দশটি স্টেটমেন্ট থাকে, তাহলে প্রথম স্টেটমেন্ট আগে মেশিন কোডে রূপান্তর হয়ে চলবে, তারপর দ্বিতীয় স্টেটমেন্ট, তারপর তৃতীয় স্টেটমেন্ট, এভাবে একে একে সব স্টেটমেন্ট এক্সিকিউট হবে। এ কারণে ভুল শুদ্ধ করা অনেক সহজ। কিন্তু একটি একটি করে স্টেটমেন্ট মেশিন কোডে রূপান্তরিত হয় বলে সময় তুলনামূলকভাবে বেশি লাগে।

৫.৪ প্রোগ্রামের সংগঠন (Program Structure)

একটি প্রোগ্রামের পুরোটির গঠন, বিশেষ করে তার ভেতরকার ছোট ছোট অংশগুলোর গঠন এবং একটির সঙ্গে অন্যটির পারস্পরিক সম্পর্ককে প্রোগ্রামের সংগঠন বা স্ট্রাকচার বলে। মাঝে মাঝেই কোনো কোনো প্রোগ্রামের সংগঠনকে ভালো এবং কোনো কোনো প্রোগ্রামের সংগঠনকে দুর্বল বলা হয়। ভালো সংগঠনের প্রোগ্রাম কিছু প্রচলিত নিয়ম মেনে চলে এবং ভিন্ন ভিন্ন অংশের ভেতরকার সম্পর্কগুলো হয় সহজ, এবং সেগুলো অনেক স্পষ্টভাবে উল্লেখ করা থাকে। সেখানে সঠিক ডেটা স্ট্রাকচার ব্যবহার করা হয় এবং প্রোগ্রামের গতি প্রবাহ (Flow Control) হয় সুনির্দিষ্ট। দুর্বল সংগঠনের প্রোগ্রামে প্রচলিত নিয়মকে উপেক্ষা করা হয় এবং বিভিন্ন অংশের ভেতরকার সম্পর্ক হয় অনিয়মিত এবং অস্পষ্ট। শুধু তাই নয় সেখানে সঠিক ডেটা স্ট্রাকচারকে গুরুত্ব দেওয়া হয় না এবং প্রোগ্রামের গতি প্রবাহ হয় এলোমেলো।

৫.৫ প্রোগ্রাম তৈরির ধাপসমূহ (Steps of Developing a Program)

প্রোগ্রাম তৈরি করার সময় শুরুতেই প্রোগ্রামাররা কোড লিখতে বসে যান না। বরং প্রথমে চিন্তা করতে হয় যে, প্রোগ্রাম লিখে যে সমস্যাটি সমাধান করা হবে, সেটি কীভাবে করা হবে। যে কাজগুলো করা হবে, সেগুলোর প্রতিটি ধাপ লিখে ফেলা হয়। এই ধাপগুলোকেই বলে অ্যালগরিদম (algorithm)। আর অনেক সময় লেখার চেয়ে ছবি এঁকে বোঝা সহজ। সমস্যা সমাধানের ধাপগুলোকে যে ছবির মাধ্যমে প্রকাশ করা হয়, তাকে বলা হয় ফ্লোচার্ট (flowchart)।

৫.৫.১ অ্যালগরিদম

ধরা যাক, একজন শিক্ষার্থী প্রতিদিন সকালে তার বাইসাইকেল চালিয়ে কলেজে যায়। তবে বাইসাইকেল যদি কোনো কারণে নষ্ট থাকে, তাহলে সে রিকশায় চড়ে কলেজে যায়। কলেজে যাওয়ার ধাপগুলোকে এভাবে লেখা যায়—

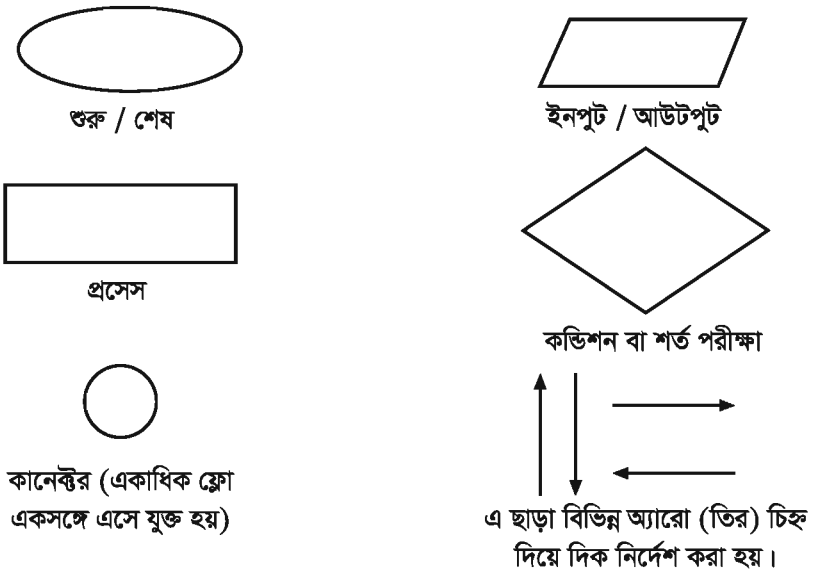
১) সাইকেল ঠিকঠাক আছে? উত্তর 'হ্যাঁ' হলে ৪নং ধাপে যাও, উত্তর 'না' হলে পরের ধাপে যাও।
২) অভিভাবকের কাছ থেকে রিকশাভাড়ার টাকা নাও।
৩) রিকশা ভাড়া করে কলেজে যাও, এরপরে পাঁচ নম্বর ধাপে যাও।
৪) সাইকেল চালিয়ে কলেজে যাও।
৫) কলেজে পৌঁছে গিয়েছ।

উপরের ধাপগুলোকে আমরা বলতে পারি ওই শিক্ষার্থীর কলেজে যাওয়ার অ্যালগরিদম। অ্যালগরিদম লেখার কোনো সুনির্দিষ্ট নিয়ম-কানুন নেই। ধাপগুলোর ক্রম সঠিক হতে হবে যেন ধাপগুলো ধারাবাহিকভাবে

অনুসরণ করলে সমস্যার সমাধান হয়। একটি ধাপের কাজ শেষ হলে তার পরের ধাপের কাজটি করতে হবে। তবে, কোনো কারণে যদি ঠিক পরের ধাপটি বাদ দিয়ে একটি বিশেষ ধাপের কাজ করতে চাই, সেক্ষেত্রে সেটি উল্লেখ করে দিতে হবে। যেমন— উপরের উদাহরণে আমরা তৃতীয় ধাপের শেষে বলে দিয়েছি, পাঁচ নম্বর ধাপে যেতে। এক্ষেত্রে চার নম্বর ধাপের কাজটি আর করা হবে না। আবার, এক নম্বর ধাপে সাইকেল যদি ঠিকঠাক থাকে, তাহলে সরাসরি চার নম্বর ধাপে চলে গিয়েছি, এক্ষেত্রে দুই এবং তিন নম্বর ধাপের কাজ আর করা হবে না।

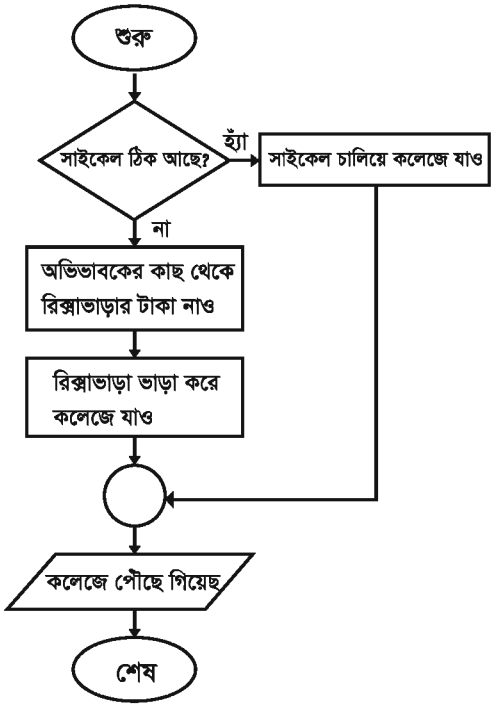
৫.৫.২ ফ্লোচার্ট

ফ্লোচার্ট তৈরির কিছু নিয়ম আছে। বিভিন্ন ধরনের নির্দেশ বোঝানোর জন্য বিভিন্ন ধরনের চিহ্ন ব্যবহার করা হয়। এরকম কিছু প্রাথমিক চিহ্ন 5.5 চিত্রে দেখানো হলো।



চিত্র 5.5 : ফ্লোচার্টের বিভিন্ন চিহ্ন

উপরের অ্যালগরিদমটি ফ্লোচার্ট আকারে প্রকাশ করা যাবে এভাবে—



চিত্র 5.6 : কলেজে যাওয়ার ফ্লোচার্ট

অ্যালগরিদম ও ফ্লোচার্ট তৈরির পরে নির্দিষ্ট প্রোগ্রামিং ভাষায় কোড লেখা হয়। কোড লেখার পরে সেই কোড বিভিন্ন টেস্ট-কেইস (test case) দিয়ে পরীক্ষা করা হয়। এক্ষেত্রে বিভিন্ন রকম ইনপুটের জন্য প্রোগ্রামটি প্রত্যাশিত আউটপুট দিচ্ছে কি না সেটি যাচাই করে দেখা হয়। যদি কোনো টেস্ট কেইসের জন্য প্রত্যাশিত আউটপুট না আসে, তখন বুঝতে হবে প্রোগ্রামটি সঠিক নয়। প্রোগ্রামটি ভুল আউটপুট দেওয়ার পেছনে দুটি কারণ থাকতে পারে। প্রথমত, সমস্যাটি সমাধানের জন্য যে অ্যালগরিদম ব্যবহার করা হয়েছে সেটি সঠিক নয়। দ্বিতীয়ত, অ্যালগরিদম সঠিক হলেও, অ্যালগরিদম অনুসরণ করে প্রোগ্রাম লেখার সময় কোনো ভুল হয়েছে। কোডে এ ধরনের ভুল থাকলে তাকে বাগ (bug) বলা হয়। এ পর্যায়ে প্রোগ্রামের যাবতীয় বাগ খুঁজে বের করে সমাধান করা হয়, অর্থাৎ, প্রোগ্রামটি ত্রুটিমুক্ত করা হয়। এই ধাপটিকে বলা হয় ডিবাগিং (debugging)।

সব টেস্ট কেইসের জন্য প্রোগ্রাম যখন ঠিকঠাক কাজ করে, তখন সেটি রিলিজ (release) করা হয়। বড় বড় প্রোগ্রামের ক্ষেত্রে রিলিজ করার সময় ব্যবহারকারীর জন্য নির্দেশনা বা ইউজার ম্যানুয়াল (User manual) তৈরি করা হয়।

প্রোগ্রাম তৈরির মূল ধাপগুলো হচ্ছে :

- যে সমস্যাটি সমাধান করা হবে, সেটিকে সঠিকভাবে বর্ণনা করা
- সমস্যার সমাধানের জন্য অ্যালগরিদম ও ফ্লোচার্ট তৈরি করা
- কোড লেখা
- প্রোগ্রাম পরীক্ষা করা ও ভুল থাকলে ডিবাগ করে প্রোগ্রাম সংশোধন করা
- প্রোগ্রাম রিলিজ করা

৫.৬ প্রোগ্রাম ডিজাইন মডেল (Program Design Model)

বাস্তব জীবনে কম্পিউটার প্রোগ্রাম কার্যকর করা যথেষ্ট সময় ও শ্রমসাপেক্ষ ব্যাপার। এর পুরো প্রক্রিয়াটি কার্যকর করার একটি গুরুত্বপুর্ণ মডেল হচ্ছে ওয়াটারফল বা জলপ্রপাত মডেল। এখানে পুরো প্রজেক্টটিকে কয়েকটি সুনির্দিষ্ট এবং ধারাবাহিক অংশে ভাগ করে নেয়া হয়। এর একটি অংশ শেষ হলেই মাত্র অন্য অংশটি শুরু করা যায়। পুরো প্রোগ্রামিং প্রক্রিয়াটি যেহেতু জলপ্রপাতের মতো একদিকে প্রবাহিত হয় সেজন্য এটিকে ওয়াটারফল বা জলপ্রপাত মডেল বলা হয়।

এই মডেল অনুযায়ী প্রোগ্রামিংয়ের ধারাবাহিক অংশগুলো হচ্ছে প্রয়োজনের বিশ্লেষণ, ডিজাইন, কোডিং, নিরীক্ষণ, কার্যক্ষেত্রে প্রতিস্থাপন এবং রক্ষণাবেক্ষণ। এই মডেলে প্রয়োজনের বিশ্লেষণে 20-40% এবং ডিজাইন ও কোডিংয়ে 30-40% সময় ব্যয় করা হয়। বাকি সময়টুকুতে নিরীক্ষণ এবং কার্যকর করার কাজে শেষ করতে হয়। এভাবে সময়ের বণ্টন যথেষ্ট যৌক্তিক, কারণ দেখা গিয়েছে প্রোগ্রামিংয়ের শুরুর দিকে একটি সমস্যার সমাধান যত সহজ, শেষের দিকে ঠিক ততটাই কঠিন। এই মডেলে ঠিকভাবে প্রোগ্রামিংয়ের তথ্য সংরক্ষণের উপর অনেক গুরুত্ব দেওয়া হয় সে কারণে একজন বা একটি টিম প্রোগ্রামিংয়ের মাঝখানে চলে গেলেও প্রোগ্রামটি সহজভাবে শেষ করা সম্ভব হয়।

৫.৫ 'সি' প্রোগ্রামিং ভাষা (Programming Language C)

এর পরের অংশটুকু পুরোপুরি ব্যাবহারিক। প্রোগ্রামিং করার ব্যবস্থা আছে (কম্পিউটারে কিংবা স্মার্টফোনে) শুধু সেরকম পরিবেশে পরের অংশটুকু শিক্ষার্থীর জন্য অর্থপূর্ণ বলে বিবেচিত হবে।

সি একটি অত্যন্ত শক্তিশালী প্রোগ্রামিং ভাষা। সি ভাষা ব্যবহার করে বিভিন্ন রকমের প্রোগ্রাম তৈরি করা যায়। যেমন—

- সিস্টেম লেভেলের প্রোগ্রাম, যা দিয়ে সরাসরি হার্ডওয়্যার নিয়ন্ত্রণ করা যায়। যেমন— কি-বোর্ড, প্রিন্টার ইত্যাদি হার্ডওয়্যার পরিচালনা করার জন্য প্রয়োজনীয় ড্রাইভার সফটওয়্যার সি ভাষা ব্যবহার করে লেখা যায়। এছাড়া যেসব ইলেকট্রনিক যন্ত্রাংশে মাইক্রোপ্রসেসর বা মাইক্রোকন্ট্রোলার থাকে, (যেমন— টেলিভিশন, রেফ্রিজারেটর, মাইক্রোওয়েভ ওভেন, ওয়াশিং মেশিন ইত্যাদি) তাতে যেসব প্রোগ্রাম তৈরি করে দেওয়া থাকে, সেখানে সি ভাষা ব্যবহার করা হয়।
- অ্যাপ্লিকেশন প্রোগ্রাম, যেগুলো ব্যবহার করে ব্যবহারকারীরা নির্দিষ্ট কোনো কাজ করতে পারে। যেমন— ছবি সম্পাদনার জনপ্রিয় সফটওয়্যার অ্যাডোবি ফটোশপ (Adobe Photoshop)।

- বিভিন্ন প্রোগ্রামিং ভাষার কম্পাইলার তৈরিতে সি ভাষা ব্যবহার করা হয়।
- কম্পিউটারের অপারেটিং সিস্টেম, যেমন— লিনাক্স (Linux) সি দিয়ে তৈরি।
- বিভিন্ন রকম ডেটাবেজ প্রোগ্রাম। ডেটাবেজ অধ্যায়ে এসকিউলাইট (SQLite) নামক যে ডেটাবেজ ম্যানেজমেন্ট সিস্টেম দেখানো হয়েছে, সেটিও সি দিয়ে তৈরি।

সি ভাষার কম্পাইলার

কম্পিউটারে সি ভাষায় প্রোগ্রাম লিখে চালাতে হলে প্রথমে ইন্টারনেট থেকে একটি কম্পাইলার সফটওয়্যার ডাউনলোড এবং ইনস্টল করে নিতে হবে। প্রথমে একটি টেক্সট ফাইলে প্রোগ্রাম লিখে সেভ করতে হবে। এই ফাইলের এক্সটেনশন হবে .c (অর্থাৎ, ফাইলটির নামের শেষে .c কথাটি থাকবে, যেমন— program1.c)। এরপরে ফাইলটি কম্পাইলার ব্যবহার করে কম্পাইল করতে হবে। কম্পাইল করার পরে একটি এক্সিকিউটেবল ফাইল তৈরি হবে। উইন্ডোজ অপারেটিং সিস্টেমে এই এক্সিকিউটেবল ফাইলের এক্সটেনশন হয় .exe (যেমন— program1.exe)।

প্রোগ্রাম কম্পাইলারগুলো সাধারণত কমান্ড লাইনভিত্তিক হয়। অর্থাৎ, কমান্ড লাইন অ্যাপ্লিকেশনে নির্দেশ টাইপ করে প্রোগ্রাম কম্পাইল করতে হয়। তবে বর্তমানে প্রোগ্রামারদের কাজ সহজ করার জন্য কিছু আইডিই সফটওয়্যার পাওয়া যায়, যেখানে, একই সঙ্গে কোড লেখা, কম্পাইল করাসহ বিভিন্ন কাজ করা যায়।

ইন্টারনেটে এরকম বিভিন্ন আইডিই (IDE) সফটওয়্যার বিনামূল্যে পাওয়া যায়। তারমধ্যে সি ভাষার জন্য ব্যবহৃত একটি আইডিই হচ্ছে কোডব্লকস (Code::blocks)। এছাড়া নেটবিনস, একলিপ্স, মাইক্রোসফট ভিজুয়াল স্টুডিওসহ বিভিন্ন সফটওয়্যার দিয়েও সি ভাষায় প্রোগ্রামিং করা যায়। এসব সফটওয়্যারের পাশাপাশি অ্যান্ড্রয়েড অপারেটিং সিস্টেম চালিত মোবাইল ফোনের জন্যও বিভিন্ন কম্পাইলার অ্যাপ পাওয়া যায়।

হ্যালো ওয়ার্ল্ড (Hello World)

এখন আমরা একটি সি প্রোগ্রাম দেখব।

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello World!");
6
7      return 0;
8  }
```

প্রোগ্রাম 5.1

কোড লেখার পরে প্রোগ্রামটি সেভ করতে হবে। সেভ করার সময় ফাইলের এক্সটেনশন দিতে হবে .c। এরপর প্রোগ্রামটি কম্পাইল এবং রান করতে হবে। প্রোগ্রামটি রান করলে আউটপুট আসবে এরকম—

```
Hello World!
```

সি ভাষায় তৈরি প্রোগ্রামে একটি নির্দিষ্ট কাজ করার জন্য একটি ফাংশন তৈরি করা হয়। ফাংশনের ভেতরে ওই কাজটি সম্পন্ন করার জন্য প্রয়োজনীয় কোড লেখা থাকে।

উপরের প্রোগ্রামের তৃতীয় লাইনে রয়েছে, int main()। একে বলা হয় main() ফাংশন। চতুর্থ এবং অষ্টম লাইনে দুটি ব্র্যাকেট (দ্বিতীয় বন্ধনী) চিহ্ন দিয়ে বোঝানো হচ্ছে main() ফাংশনটি চতুর্থ লাইনে শুরু হয়েছে এবং অষ্টম লাইনে শেষ হয়েছে। পঞ্চম ও সপ্তম লাইনে দুটি নির্দেশ দেওয়া হয়েছে। আর ষষ্ঠ লাইনটি ফাঁকা রাখা হয়েছে।

সি ভাষায় লেখা যে কোনো প্রোগ্রাম চলা শুরু হয় main() ফাংশন থেকে। যেমন— উপরের কোডে তৃতীয় লাইনে main() ফাংশন থেকে এই প্রোগ্রামটি চলতে আরম্ভ করবে। একটি প্রোগ্রামে কেবল একটি main() ফাংশনই লেখা হয়।

এর পরে পঞ্চম লাইনে রয়েছে printf ("Hello World!") স্টেটমেন্ট। printf() একটি ফাংশন, যার কাজ হচ্ছে স্ক্রিনে কোনো কিছু প্রিন্ট করা। যেমন— এই প্রোগ্রামের ক্ষেত্রে এই স্টেটমেন্টটি স্ক্রিনে Hello World! কথাটি প্রিন্ট করছে। printf() ফাংশনটি কীভাবে প্রিন্ট করার কাজটি করবে, সেটি এই প্রোগ্রামে কোথাও বলা নেই, তবে stdio.h নামক একটি ফাইলে বলা আছে। একে বলে হেডার (header) ফাইল। হেডার ফাইলে বিভিন্ন ফাংশন তৈরি করে দেওয়া থাকে। এই ফাংশনগুলো ব্যবহার করার জন্য হেডার ফাইলটি প্রোগ্রামে অন্তর্ভুক্ত করতে হয়।

প্রথম লাইনে #include <stdio.h> লেখার কারণে stdio.h ফাইলে যে সব ফাংশন দেওয়া আছে, সেগুলো এই প্রোগ্রামে ব্যবহার করা যাবে। stdio.h হেডার ফাইলে ব্যবহারকারীর কাছ থেকে ইনপুট নেওয়া ও আউটপুট প্রিন্ট করা সংক্রান্ত বেশ কিছু ফাংশন লেখা আছে।

প্রোগ্রামের সপ্তম লাইনে লেখা আছে, return 0। এটি মেইন ফাংশনের শেষ লাইন। এই লাইনটি কী কাজ করে তা নিয়ে এ অধ্যায়ের পরবর্তী অংশে আলোচনা করা হয়েছে। এই লাইনটি চলার পরে এই প্রোগ্রামটি চলা শেষ হবে।

নিজে করি ১ : একটি প্রোগ্রাম লিখতে হবে, যেটি স্ক্রিনে I love Bangladesh. কথাটি প্রিন্ট করবে।

ডেটা টাইপ (Types of Data)

আমরা জানি কম্পিউটার প্রসেসর বিভিন্ন হিসাব-নিকাশ করে। এই হিসাব-নিকাশগুলো বিভিন্ন ডেটার উপরে করা হয়। সি প্রোগ্রামিং ভাষায় বেশ কিছু ডেটা টাইপ রয়েছে, অর্থাৎ বিভিন্ন ধরনের ডেটা নিয়ে কাজ করার ব্যবস্থা রয়েছে। এর মধ্যে উল্লেখযোগ্য হচ্ছে— char, int, float ও double। নিচে ডেটা টাইপগুলো নিয়ে সংক্ষিপ্ত আলোচনা করা হলো—

char

এটি হচ্ছে character-এর প্রথম চারটি অক্ষর। এ ধরনের ডেটা টাইপ একটিমাত্র অক্ষর (বর্ণ, অংক, যতিচিহ্ন ইত্যাদি) ধারণ করতে পারে, যেমন— 'a', 'D', '5', '!' ইত্যাদি। এটি কম্পিউটারের মেমোরিতে সাধারণত এক বাইট (অর্থাৎ, আট বিট) জায়গা দখল করে। তাহলে এ ধরনের ডেটা টাইপে 2^8 বা 256টি পৃথক ডেটা রাখা যায়। 256টি জিনিস কিন্তু একটি ভ্যারিয়েবলে একসঙ্গে রাখা যায় না, একটি ভ্যারিয়েবলে একই সময়ে কেবল একটি ডেটা রাখা যায়, আর char টাইপের ডেটার ক্ষেত্রে সম্ভাব্য 256টি মানের যে কোনো একটি রাখা যায়।

একটি বিটে যে কোনো সময়ে রাখা যায় 0 অথবা, 1। অর্থাৎ, একটি বিট দিয়ে দুটি ভিন্ন জিনিস প্রকাশ করা যায়। আবার দুইটি বিট দিয়ে প্রকাশ করা যায় চারটি ভিন্ন জিনিস— 00, 01, 10, এবং 11। একইভাবে তিনটি বিট দিয়ে প্রকাশ করা যায় আটটি ভিন্ন জিনিস— 000, 001, 010, 011, 100, 101, 110, এবং 111। তাহলে n-সংখ্যক বিট দিয়ে প্রকাশ করা যায়, 2^n-সংখ্যক ভিন্ন জিনিস।

int

Integer শব্দের অর্থ পূর্ণসংখ্যা। এই শব্দের প্রথম তিনটি অক্ষর নিয়ে int ডেটা টাইপের নামকরণ করা হয়েছে। এ ধরনের ডেটা টাইপে পূর্ণসংখ্যা রাখা যায়। একটি int টাইপের ডেটা সাধারণত কম্পিউটারের মেমোরিতে চার বাইট (অর্থাৎ, 32 বিট) জায়গা দখল করে। যেহেতু এর সাইজ 32 বিট, তাই এতে সম্ভাব্য 2^{32} বা 4294967296 রকমের সংখ্যা রাখা যায়। আর সংখ্যা যেহেতু ধনাত্মক ও ঋণাত্মক উভয় জাতীয় হতে পারে, তাই -2147483648 থেকে 2147483647 সীমার মধ্যে যে কোনো সংখ্যা int টাইপের ডেটাতে ধারণ করা যায়।

float

দশমিকযুক্ত সংখ্যা অর্থাৎ, floating point number রাখার জন্য float ডেটা টাইপ ব্যবহার করা হয়। এটি মেমোরিতে সাধারণত চার বাইট জায়গা দখল করে।

double

এটিও দশমিক যুক্ত সংখ্যা রাখার জন্য ব্যবহার করা হয়। তবে এটি সাধারণত কম্পিউটারের মেমোরিতে আট বাইট জায়গা নেয়।

কয়েকটি প্রোগ্রাম লিখে উল্লিখিত ডেটা টাইপের ব্যবহার দেখানো হলো—

উদাহরণ ১

```
#include <stdio.h>
int main()
{
    char ch;
    ch = 'X';
    printf("The character is %c", ch);
    return 0;
}
```

প্রোগ্রাম 5.2

প্রোগ্রামটি কম্পাইল ও রান করলে আউটপুট আসবে The character is X।

এই প্রোগ্রামে char ch লিখে char টাইপের একটি ভ্যারিয়েবল (variable) তৈরি করা হয়েছে, যার নাম দেওয়া হয়েছে ch। এখানে ch-এর বদলে অন্য নামও ব্যবহার করা যেত। একে ভ্যারিয়েবল বলা হয়। ক্যারেক্টার টাইপের ডেটা প্রিন্ট করার জন্য %c ব্যবহার করা হয়। একে বলা হয় ফরম্যাট স্পেসিফায়ার (format specifier)। নিচের টেবিলে বিভিন্ন ডেটা টাইপের ফরম্যাট স্পেসিফায়ার দেখানো হলো—

ডেটা টাইপ	ফরম্যাট স্পেসিফায়ার
char	%c
int	%d
float	%f
double	%lf (এখানে l হচ্ছে ছোট হাতের ইংরেজি L অক্ষর)

সি প্রোগ্রামে ভ্যারিয়েবলের নাম লেখার ক্ষেত্রে কিছু নিয়মকানুন রয়েছে। ভ্যারিয়েবলের নামে কেবল বর্ণ, অংক এবং বিশেষ চিহ্ন ব্যবহার করা যাবে। তবে নামের প্রথম অক্ষরটি কোনো অংক হতে পারবে না। Variable এ underscore (_) এবং doller সাইন ($) ছাড়া অন্য কোনো স্পেশাল ক্যারেক্টার (@, !, %, +, - ...ইত্যাদি) এবং punctuation character (,.: ইত্যাদি) বা কোনো ধরনের অপারেটর ব্যবহার করা যাবে না। Variable নামে কোনো স্পেস দেয়া যাবে না। C তে uppercase and lowercase character এর মধ্যে পার্থক্য আছে যার কারণে C কে case sensitive programming language বলা হয়। তাই aaa, AAA, Aaa ba Net, net, NeT ইত্যাদি variable এর মান এক নয়। স্পেশালি কোনো কী-ওয়ার্ডকে variable এর নাম হিসেবে ব্যবহার করা যাবে না।

সঠিক ভ্যারিয়েবল নামের উদাহরণ	ভুল ভ্যারিয়েবল নামের উদাহরণ
age final_result student_1_marks student0 __current_date	0student final result greetings! my,name

উদাহরণ ২

একটি ক্যারেক্টার টাইপের ভ্যারিয়েবল char টাইপের যে কোনো ডেটা ধারণ করতে পারে। বিষয়টি একটি প্রোগ্রাম লিখে দেখা যাক।

```
#include <stdio.h>
int main()
{
    char ch;
    ch = 'x';
    printf("Value stored in ch is %c\n", ch);
    ch = 'y';
    printf("Value stored in ch is %c\n", ch);
    return 0;
}
```

প্রোগ্রাম 5.3

উপরের প্রোগ্রামটি কম্পাইল করে রান করলে নিচের মতো আউটপুট পাওয়া যাবে।

```
Value stored in ch is x
Value stored in ch is y
```

আউটপুট দেখে বোঝার চেষ্টা করতে হবে যে প্রোগ্রামটিতে কী কাজ হচ্ছে। এখানে printf() ফাংশনের ভেতরে \n ব্যবহার করা হয়েছে। \n-এর মানে হচ্ছে নিউ লাইন (new line), অর্থাৎ এটি প্রিন্ট করলে আউটপুটের পরবর্তী অংশ স্ক্রিনে নতুন লাইনে চলে যাবে। যদি printf() ফাংশনের ভেতরে \n ব্যবহার করা না হতো, তবে আউটপুট হতো এরকম—

```
Value stored in ch is x Value stored in ch is y
```

একটি ভ্যারিয়েবলে যখন কোনো মান রাখা হয় (যেমন— ch = 'x';), একে বলা হয় ch-এর মধ্যে 'x' অ্যাসাইন করা এবং অপারেশনটির নাম হচ্ছে অ্যাসাইনমেন্ট অপারেশন। একটি ভ্যারিয়েবলে একই সময়ে কেবল একটি মান অ্যাসাইন করা যায়।

নিজে করি ২ : প্রোগ্রাম 5.3-এ ch-এর বদলে বিভিন্ন নামের ভ্যারিয়েবল ব্যবহার করে পরীক্ষা-নিরীক্ষা করতে হবে। প্রতিবার প্রোগ্রাম সেভ করে কম্পাইল ও রান করতে হবে।

উদাহরণ ৩

এখন একটি প্রোগ্রাম দেখানো হবে, যার কাজ হচ্ছে দুটি সংখ্যার যোগফল, বিয়োগফল, গুণফল ও ভাগফল প্রকাশ করা—

```
#include <stdio.h>
int main()
{
    int number1, number2;
    number1 = 12;
    number2 = 4;
    printf("number1 + number2 = %d\n", number1 + number2);
    printf("number1 - number2 = %d\n", number1 - number2);
    printf("number1 * number2 = %d\n", number1 * number2);
    printf("number1 / number2 = %d\n", number1 / number2);
    return 0;
}
```

প্রোগ্রাম 5.4

আউটপুট

```
number1 + number2 = 16
number1 - number2 = 8
number1 * number2 = 48
number1 / number2 = 3
```

ইন্টিজার টাইপের ডেটা প্রিন্ট করার জন্য %d ব্যবহার করা হয়। আর গুণচিহ্ন ও ভাগচিহ্ন হচ্ছে, যথাক্রমে * ও /। উপরের প্রোগ্রামটিতে চাইলে আরেকটি ভ্যারিয়েবল তৈরি করা যায়, যেখানে বিভিন্ন গাণিতিক অপারেশনের ফলাফল রাখা হবে।

```
#include <stdio.h>
int main()
{
    int number1, number2, result;
    number1 = 12;
    number2 = 4;
    result = number1 + number2;
    printf("number1 + number2 = %d\n", result);
    result = number1 - number2;
    printf("number1 - number2 = %d\n", result);
    result = number1 * number2;
    printf("number1 * number2 = %d\n", result);
    result = number1 / number2;
    printf("number1 / number2 = %d\n", result);
    return 0;
}
```

প্রোগ্রাম 5.5

উল্লেখ্য যে, number1 + number2 হচ্ছে একটি এক্সপ্রেশন (expression)। সি প্রোগ্রামিংয়ে এক্সপ্রেশন বলতে বোঝানো হয় কিছু কোড যা একটি মান প্রকাশ করে। আবার number2 = 4; হচ্ছে একটি স্টেটমেন্ট (statement)। একটি স্টেটমেন্ট দিয়ে একটি কাজ বোঝানো হয়। এখানে কাজটি হচ্ছে number2 নামক ভ্যারিয়েবলে 4 রাখা। আবার, result = number1 + number2; একটি স্টেটমেন্ট। এটি দ্বারা বোঝানো হচ্ছে number1 + number2 এক্সপ্রেশনটি এক্সিকিউট করে যে মান পাওয়া যাবে, সেটি result নামক ভ্যারিয়েবলে রাখা। printf("number1 / number2 = %d\n", result); - এটিও একটি স্টেটমেন্ট। প্রতিটি স্টেটমেন্টের শেষে একটি সেমিকোলন চিহ্ন দেওয়া হয়।

কি-ওয়ার্ড (Keyword)

সি প্রোগ্রামিং ভাষায় বেশ কিছু সংরক্ষিত শব্দ আছে, যেগুলো হচ্ছে সি ভাষার অংশ। তাই এসব শব্দকে ভ্যারিয়েবলের নাম কিংবা ফাংশনের নাম হিসেবে ব্যবহার করা যায় না। এসব শব্দকে বলা হয় কি-ওয়ার্ড। নিচের টেবিলে C ভাষার ANSI (ANSI: American National Standard Institute) কি-ওয়ার্ডের একটি তালিকা দেওয়া হলো :

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

এ সমস্ত কি-ওয়ার্ড মুখস্থ করার প্রয়োজন নেই। কেবল খেয়াল রাখতে হবে, এই নামগুলো ভ্যারিয়েবল কিংবা ফাংশনের নাম হিসেবে ব্যবহার করা যাবে না।

ইনপুট আউটপুট স্টেটমেন্ট (Input Output Statements)

ইতোমধ্যে দেখানো হয়েছে যে, কীভাবে স্ক্রিনে প্রিন্ট করতে হয়, অর্থাৎ, আউটপুট দিতে হয়। এবারে ইনপুট নেওয়ার পালা। নিচের প্রোগ্রামটি ব্যবহারকারীর কাছ থেকে দুটি সংখ্যা ইনপুট নেবে এবং তাদের যোগফল আউটপুটে দেখাবে।

উদাহরণ ৪

```
#include <stdio.h>
int main()
{
    int n1, n2;
    scanf("%d %d", &n1, &n2);
    printf("%d\n", n1+n2);
    return 0;
}
```

প্রোগ্রাম 5.6

প্রোগ্রামটি কম্পাইল করে রান করলে সেটি ব্যবহারকারীর ইনপুটের জন্য অপেক্ষা করবে (চিত্র 5.7), দুটি সংখ্যা লিখে কি-বোর্ডের এন্টার কি (key) চাপলে তখন আউটপুট দেখাবে (চিত্র 5.8)।

চিত্র 5.7 : কমান্ড লাইনে কিছু প্রিন্ট না করে ব্যবহারকারীর ইনপুটের জন্য অপেক্ষা করছে

চিত্র 5.8 : দুটি সংখ্যা টাইপ করে কি-বোর্ডে Enter কি চাপার পরে ফলাফল পাওয়া গেল

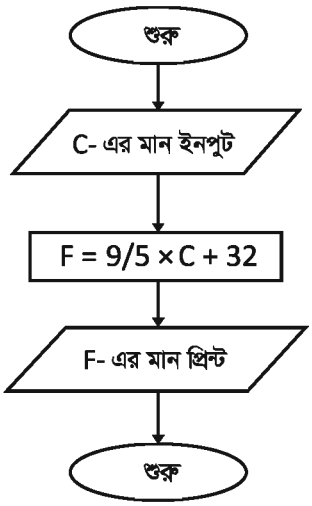
তাহলে দেখা যাচ্ছে, scanf() ফাংশনটি দিয়ে ইনপুট নেওয়া হয়। আর যেসব ভ্যারিয়েবলের মান ইনপুট নেওয়া হচ্ছে, তাদের আগে অ্যামপারসেন্ড (&) চিহ্ন ব্যবহার করা হয়। আর ফাংশনটির ভেতরে printf() ফাংশনের মতো একই ফরম্যাট স্পেসিফায়ার ব্যবহার করা হয়।

উদাহরণ ৫

একটি প্রোগ্রাম লিখতে হবে, যা কোনো তাপমাত্রাকে সেলসিয়াস এককে ইনপুট নেবে এবং ফারেনহাইট এককে আউটপুট দেবে। প্রোগ্রাম লেখার আগে প্রোগ্রামটির ফ্লোচার্ট তৈরি করতে হবে।

সেলসিয়াস থেকে ফারেনহাইটে রূপান্তর করার সূত্র হচ্ছে, $\frac{C}{5} = \frac{F-32}{9}$ সুতরাং $F = 9/5 \times C + 32$

ফ্লোচার্টটি 5.9 চিত্রে দেখানো হলো।



চিত্র 5.7 : সেলসিয়াস থেকে ফারেনহাইট রূপান্তরের ফ্লোচার্ট

আর প্রোগ্রামটি হবে এরকম—

```
#include <stdio.h>
int main()
{
    double C, F;
    scanf("%lf", &C);
    F = 9/5 * C + 32;
    printf("%lf\n", F);
    return 0;
}
```

প্রোগ্রাম 5.7

নিজে করি ৩ : একটি প্রোগ্রাম লিখতে হবে, যা কোনো তাপমাত্রাকে ফারেনহাইট এককে ইনপুট নেবে এবং সেলসিয়াস এককে আউটপুট দেবে।

কনডিশনাল স্টেটমেন্ট (Conditional Statements)

কম্পিউটার প্রোগ্রাম এমনভাবে লেখা যায়, যেন প্রোগ্রামটি বিভিন্ন শর্তের উপর ভিত্তি করে সিদ্ধান্ত গ্রহণ করতে পারে। এই শর্তকে প্রোগ্রামিংয়ের ভাষায় বলে কন্ডিশন (condition), আর যে এক্সপ্রেশন ব্যবহার করে শর্ত তৈরি করা হয়, তাকে বলে কন্ডিশনাল এক্সপ্রেশন (conditional expression)। কন্ডিশনাল এক্সপ্রেশন যখন এক্সিকিউট হয়, তখন তার ফলাফল হবে হয় সত্য (True), নয়তো মিথ্যা (False)।

রিলেশনাল অপারেটর (Relational Operator)

সি প্রোগ্রামিং ভাষায়, দুটি সংখ্যা তুলনা করার জন্য ছয়টি অপারেটর আছে, এগুলোকে বলা হয় রিলেশনাল অপারেটর। নিচের টেবিলে সেগুলো দেখানো হলো—

অপারেটর	ব্যাখ্যা
==	দুটি সংখ্যা সমান কি না সেটি পরীক্ষা করা হয়। সমান হলে ফলাফল সত্য আর সমান না হলে ফলাফল মিথ্যা হয়।
!=	দুটি সংখ্যা অসমান কি না সেটি পরীক্ষা করা হয়। অসমান হলে ফলাফল সত্য আর সমান হলে ফলাফল মিথ্যা হয়।
>	দুটি সংখ্যার মধ্যে বামপক্ষ ডানপক্ষের চেয়ে বড় কি না সেটি পরীক্ষা করা হয়, বামপক্ষ যদি বড় হয় তাহলে ফলাফল সত্য, আর তা না হলে ফলাফল মিথ্যা।
>=	দুটি সংখ্যার মধ্যে বামপক্ষ ডানপক্ষের চেয়ে বড় অথবা সমান কি না সেটি পরীক্ষা করা হয়, বামপক্ষ যদি বড় হয়, অথবা ডানপক্ষের সমান হয় তাহলে ফলাফল সত্য, আর তা না হলে ফলাফল মিথ্যা।
<	দুটি সংখ্যার মধ্যে বামপক্ষ ডানপক্ষের চেয়ে ছোট কি না সেটি পরীক্ষা করা হয়, বামপক্ষ যদি ছোট হয় তাহলে ফলাফল সত্য, আর তা না হলে ফলাফল মিথ্যা।
<=	দুটি সংখ্যার মধ্যে বামপক্ষ ডানপক্ষের চেয়ে ছোট অথবা সমান কি না সেটি পরীক্ষা করা হয়, বামপক্ষ যদি ছোট হয়, অথবা ডানপক্ষের সমান হয়, তাহলে ফলাফল সত্য, আর তা না হলে ফলাফল মিথ্যা।

টেবিল **5.1**

if স্টেটমেন্ট

সি প্রোগ্রামিং ভাষায় বিভিন্ন শর্ত পরীক্ষার জন্য if স্টেটমেন্ট ব্যবহার করা হয়।

```
if (conditional expression)
{
    statement 1;
    ....
}
```

প্রথম বন্ধনীর ভেতরের conditional expression যদি সত্য হয়, তাহলে if ব্লকের ভেতরের কাজ হবে। কন্ডিশনাল এক্সপ্রেশনের পরে দ্বিতীয় বন্ধনী দিয়ে ব্লকটি আবদ্ধ থাকে। ব্লকের ভেতরে এক বা একাধিক স্টেটমেন্ট থাকতে পারে।

উদাহরণ ৬

এখন একটি প্রোগ্রাম লেখা হবে, যেটি দুটি সংখ্যার মধ্যে তুলনা করে বের করবে তারা সমান কি না।

```
#include <stdio.h>
int main()
{
    int n1 = 5, n2 = 7;
    if (n1 == n2)
    {
        printf("Numbers are equal.");
    }
    return 0;
}
```

প্রোগ্রাম 5.8

প্রোগ্রামটি রান করলে আমরা আউটপুটে কিছুই দেখতে পাব না। কেননা, if-এর ভেতরে ব্যবহৃত কন্ডিশনাল এক্সপ্রেশনটির মান মিথ্যা। তাই if-ব্লকের ভেতরের কোড এক্সিকিউট হয়নি।

if-else স্টেটমেন্ট : Conditional expression সঠিক হলে এক ধরনের কাজ এবং conditional expression সঠিক না হলে অন্য ধরনের কাজ সম্পাদন করতে হয় সেক্ষেত্রে if else স্টেটমেন্ট ব্যবহৃত হয়।

```
if (conditinal expression)
{
   statement, if condition is true;
   }
   else
   {
   statement, if condition is false;
}
```

```
#include <stdio.h>
int main()
{
    int n1 = 5, n2 = 7;
    if (n1 == n2)
    {
        printf("Numbers are equal.");
    }
    else
    {
        printf("Numbers are not equal.");
    }
    return 0;
}
```

প্রোগ্রাম 5.9

এই প্রোগ্রামটি রান করলে আমরা এরকম আউটপুট দেখতে পাব—

```
Numbers are not equal.
```

if-এর সঙ্গে ব্যবহৃত কন্ডিশনাল এক্সপ্রেশনটি (n1 == n2) মিথ্যা হওয়ায় if ব্লকের কোড এক্সিকিউট হয়নি, else ব্লকের কোডগুলো এক্সিকিউট হয়েছে। আবার যদি, n1, এবং n2 দুটি ভ্যারিয়েবলের মান সমান হতো, তাহলে if ব্লকের কোড এক্সিকিউট হতো, কিন্তু else ব্লকের কোড এক্সিকিউট হতো না। তখন আমরা আউটপুট দেখতাম—

```
Numbers are equal.
```

আবার আরেকটি ব্লক আছে, else if. কখনো যদি এমন প্রোগ্রাম লেখা হয় যে একটি শর্তের পরে অন্য একটি শর্ত পরীক্ষা করা হবে, তখন if-এর সঙ্গে এক বা একাধিক else if ব্লক ব্যবহার করা হয়। যেমন—

else if chain স্টেটমেন্ট : conditional expression যদি একাধিক হয় তাহলে else if স্টেটমেন্ট ব্যবহৃত হয়।

```
if (conditional  expression 1)
{
    statement, if conditional expression-1 is true ;
    }
     else if (conditional expression-2)
    {
    statement, if conditional expression-2 is true,
    }
. . . . . .
. . . . . .
    else
    {
    statement, if both conditions are false ;
}
```

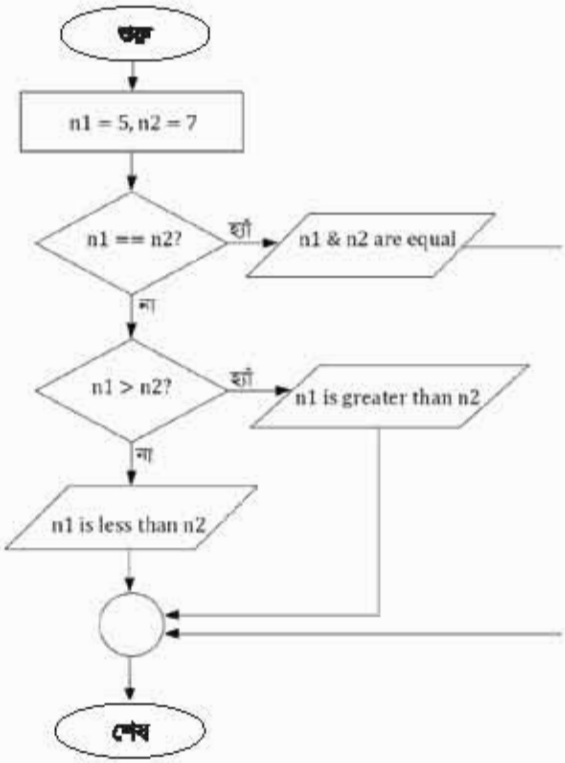
```
#include <stdio.h>
int main()
{
    int n1 = 5, n2 = 7;
    if (n1 == n2)
    {
        printf("Numbers are equal.");
    }
    else if (n1 > n2)
    {
        printf("n1 is greater than n2.");
    }
    else
    {
        printf("n1 is smaller than n2.");
    }
    return 0;
}
```

প্রোগ্রাম 5.10

এক্ষেত্রে যে ব্লকের কন্ডিশনাল এক্সপ্রেশনটি সত্য শুধু সেই ব্লকটির কোড এক্সিকিউট হবে, অন্য কোনো ব্লকের

কোড এক্সিকিউট হবে না। আর যদি কোনো ব্লকের শর্তই সত্য না হয়, তাহলে, সবশেষের else ব্লকের কোড এক্সিকিউট হবে।

উপরের প্রোগ্রামটির যদি ফ্লোচার্ট তৈরি করা হয়, সেটি হবে চিত্র 5.10 এর মত।



চিত্র 5.8: দুটি সংখ্যার মধ্যে তুলনা করার ফ্লোচার্ট

এখন আরেকটি উদাহরণ দেখানো হবে।

উদাহরণ ৭

ধরা যাক, কোনো পরীক্ষায় একজন শিক্ষার্থীর প্রাপ্ত নম্বর ইনপুট নেওয়া হবে। এই নম্বরের উপর ভিত্তি করে এই বিষয়ের লেটার গ্রেড আউটপুট দেখানো হবে।

```
#include <stdio.h>
int main()
{
    int marks;
    scanf("%d", &marks);
    if (marks >= 80){
        printf("Your grade is A+\n");
    }
    else if (marks >= 70){
        printf("Your grade is A\n");
```

```
    }
    else if (marks >= 60){
        printf("Your grade is A-\n");
    }
    else if (marks >= 50){
        printf("Your grade is B\n");
    }
    else if (marks >= 40){
        printf("Your grade is C\n");
    }
    else if (marks >= 33){
        printf("Your grade is D\n");
    }
    else{
        printf("Your grade is F\n");
    }
    return 0;
}
```

প্রোগ্রাম 5.11

এভাবে অসংখ্য if, else if যখন পরপর থাকে, তখন কোনো একটি শর্ত যদি সত্য হয়, তখন বাকি else if গুলোর শর্ত আর পরীক্ষা করা হয় না। যেমন— ইনপুট যদি হয় 75, তখন প্রথমে marks >= 80 শর্তটি পরীক্ষা করা হবে। শর্তটি মিথ্যা, তাই পরবর্তী শর্ত (marks >= 70) পরীক্ষা করা হবে। এটি সত্য। তাই এই ব্লকের ভেতরের কাজ শুরু হয়ে যাবে। এক্ষেত্রে printf() স্টেটমেন্টটি এক্সিকিউট হবে। তারপরে কিন্তু আর কোনো else if ব্লকের শর্ত পরীক্ষা করা হবে না।

নিজে করি ৪ : উপরের প্রোগ্রামে নিচের সংখ্যা ইনপুট দেওয়া হলে কী আউটপুট পাওয়া যাবে তা নির্ণয় করি :

ক) 98
খ) 80
গ) 79
ঘ) 64
ঙ) 37
চ) 23
ছ) -20

লজিকাল অপারেটর (Logical Operator)

একাধিক শর্ত মিলিয়ে নতুন শর্ত তৈরি করার জন্য গাণিতিক এক্সপ্রেশনের মতো, লজিকাল এক্সপ্রেশন লেখা যায়। বিভিন্ন শর্ত লজিকাল অপারেটর দিয়ে যুক্ত করে লজিকাল এক্সপ্রেশন তৈরি করা হয়।

সি প্রোগ্রামিং ভাষায় তিন ধরনের লজিক্যাল অপারটের আছে— && (and), || (or) এবং ! (not) অপারেটর।

অ্যান্ড (&&) অপারেটরের ক্ষেত্রে, বাম পক্ষে একটি শর্ত ও ডান পক্ষে একটি শর্ত থাকবে। যদি দুটি শর্তই সত্য হয়, তাহলে পুরো এক্সপ্রেশনটি সত্য হবে। যে কোনো একটি বা দুটি শর্তই যদি মিথ্যা হয়, তাহলে পুরো শর্তটি মিথ্যা হবে।

A	B	A && B
True	True	True
True	False	False
False	True	False
False	False	False

টেবিল **5.2**

অর (||) অপারেটরের ক্ষেত্রে, বাম পক্ষে একটি শর্ত ও ডান পক্ষে একটি শর্ত থাকবে। যদি দুটি শর্তের কমপক্ষে একটি সত্য হয়, তাহলে || সহ পুরো শর্তটি সত্য হবে। দুটি শর্তই যদি মিথ্যা হয়, তাহলে পুরো শর্তটি মিথ্যা হবে।

A	B	A || B
True	True	True
True	False	True
False	True	True
False	False	False

টেবিল **5.3**

নট (!) অপারেটরের বেলায়, অপারেটরের পরে কেবল একটি শর্ত থাকবে। শর্তটি সত্যি হলে পুরো শর্তটি মিথ্যা, আর শর্তটি মিথ্যা হলে পুরো শর্তটি সত্য হবে।

A	!A
True	False
False	True

টেবিল **5.4**

উদাহরণ ৮

ধরা যাক, কোনো একটি চাকরির আবেদনকারীদের বয়সসীমা নির্ধারণ করা হলো 18 থেকে 35। এখন একটি প্রোগ্রাম লিখতে হবে, যেটি আবেদনকারীর বয়স ইনপুট নেবে এবং বয়সের হিসেবে যে আবেদন করার যোগ্য কি না, সেটি প্রিন্ট করবে।

```
#include <stdio.h>
int main()
{
    int age;
    scanf("%d", &age);
```

```
    if (age >= 18 && age <= 35)
    {
        printf("Yes, you are eligible.\n");
    }
    else
    {
        printf("Sorry, you are not eligible.\n");
    }
    return 0;
}
```

প্রোগ্রাম 5.12

উপরের প্রোগ্রামটিতে if ব্লকের ভেতরে দুটি শর্ত ব্যবহার করা হয়েছে এবং শর্ত দুটি && অপারেটর দ্বারা যুক্ত করা হয়েছে। অর্থাৎ, এক্ষেত্রে age >= 18 এবং age <= 35 দুটি শর্তই যদি সত্য হয়, তাহলে পুরো শর্তটি সত্য হবে। প্রোগ্রামটিতে যদি !(age < 18 || age > 35) শর্ত ব্যবহার করা হতো, তাহলেও প্রোগ্রামটি একই কাজ করত।

উদাহরণ ৯

একটি সংখ্যা ইনপুট নেওয়া হবে। সংখ্যাটি 3 দ্বারা বিভাজ্য হলে Fizz প্রিন্ট করতে হবে, 5 দ্বারা বিভাজ্য হলে Buzz প্রিন্ট করতে হবে, আর সংখ্যাটি যদি 3 ও 5 উভয় সংখ্যা দ্বারাই বিভাজ্য হয়, তাহলে প্রিন্ট করতে হবে FizzBuzz.

কোনো সংখ্যা a যদি b দ্বারা বিভাজ্য হয়, তাহলে ভাগশেষ থাকবে 0। সি প্রোগ্রামিং ভাষায় ভাগশেষ বের করার অপারেটর হচ্ছে % (একে বলে মডুলাস modulus অপারেটর)। a % b-এর মান 0 হলে b দ্বারা a বিভাজ্য।

```
#include <stdio.h>
int main()
{
    int num;
    scanf("%d", &num);
    if (num % 3 == 0 && num % 5 == 0)
    {
        printf("FizzBuzz\n");
    }
    else if (num % 3 == 0)
    {
        printf("Fizz\n");
    }
    else if (num % 5 == 0)
    {
        printf("Buzz\n");
    }
    return 0;
}
```

প্রোগ্রাম 5.13

নিজে করি ৫ : উপরের প্রোগ্রামটির ফ্লোচার্ট তৈরি করি।

লুপ স্টেটমেন্ট (Loop Statements)

একই কাজ বারবার করার জন্য প্রোগ্রামিং ভাষায় লুপ স্টেটমেন্ট থাকে। সি প্রোগ্রামিং ভাষায় তিন ধরনের লুপ আছে, while লুপ, do-while লুপ এবং for লুপ।

while লুপ

while লুপের সিনট্যাক্স হচ্ছে—

```
while (condition)
{
    statement;
    ...
}
```

এখানে condition সত্য হলে, while-এর ব্লকের ভেতরের কাজ করা হবে। কাজ শেষে আবার condition পরীক্ষা করা হবে। এবারেও condition সত্য হলে আবারো while-এর ব্লকের ভেতরের কাজ করা হবে। এভাবে চক্রাকারে কাজটি বারবার চলতে থাকবে যতক্ষণ পর্যন্ত condition সত্য থাকে। যেমন— ধরা যাক, একটি প্রোগ্রাম লিখতে হবে, যেটি I Love Bangladesh কথাটি পাঁচবার প্রিন্ট করবে।

উদাহরণ ১০

```
#include <stdio.h>
int main()
{
    int i;
    i = 0;
    while (i < 5) {
        printf("I Love Bangladesh.\n");
        i = i + 1;
  }
    return 0;
}
```

প্রোগ্রাম 5.14

i = 0; স্টেটমেন্টে i তে 0 রাখা হয়েছে। তারপর while-এর ভেতরের শর্ত পরীক্ষা করা হবে। i < 5 শর্তটি সত্য, কারণ i-এর মান এখন 0। তারপর printf() স্টেটমেন্টের কাজ হবে। তারপর i = i + 1; স্টেটমেন্টটি এক্সিকিউট হবে। এই স্টেটমেন্টে i-এর মানের সঙ্গে 1 যোগ করে সেটি আবার i-তে রাখা হয়েছে (বা অ্যাসাইন করা হয়েছে)।

i-এর মান এখন 1। তারপরে আবার i < 5 শর্তটি পরীক্ষা করা হবে এবং এবারো শর্তটি সত্য (i-এর মান এখন 1)। তাই printf() ফাংশনটি এক্সিকিউট হবে। তারপরে আবার i-এর মান 1 বাড়বে। এভাবে চলতে থাকবে এবং যখন i-এর মান বেড়ে 5 হবে, তখন i < 5 শর্তটি মিথ্যা হয়ে যাবে এবং প্রোগ্রামটি while লুপের বাইরে চলে আসবে। i-এর পাঁচটি মান (0, 1, 2, 3, 4)-এর জন্য printf() ফাংশনটি পাঁচবার এক্সিকিউট হবে এবং পাঁচবার I Love Bangladesh কথাটি প্রিন্ট হবে।

নিজে করি ৬ : কথাটি একশবার প্রিন্ট করতে চাইলে কী করতে হবে?

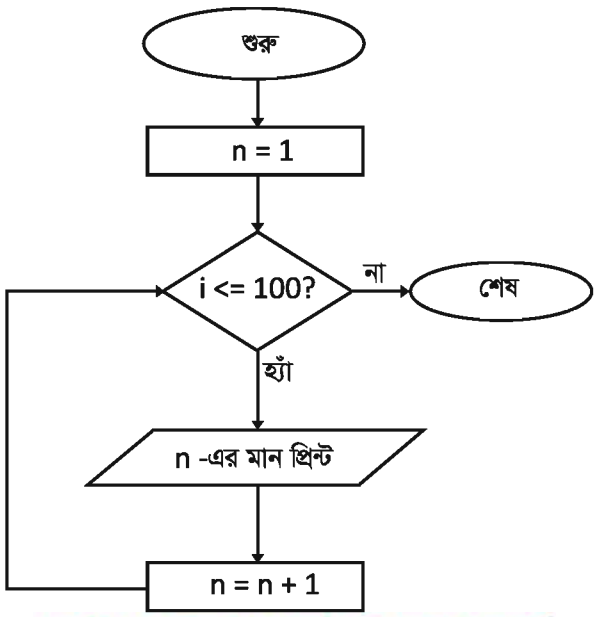
উদাহরণ ১১

এখন আরেকটি প্রোগ্রাম লেখা হবে, যার কাজ হবে 1 থেকে 100 পর্যন্ত সব সংখ্যা প্রিন্ট করা।

```
#include <stdio.h>
int main()
{
    int n;
    n = 1;
    while (n <= 100) {
        printf("%d\n", n);
        n = n + 1;
    }
    return 0;
}
```

প্রোগ্রাম 5.15

1 থেকে 100 পর্যন্ত প্রতিটি সংখ্যা প্রিন্ট করার প্রোগ্রামের ফ্লোচার্ট 5.11 চিত্রে দেখানো হল।



চিত্র 5.9 : 1 থেকে 100 পর্যন্ত প্রিন্ট করার ফ্লোচার্ট

উদাহরণ ১২

এখন, 1 থেকে 100 পর্যন্ত সব জোড় সংখ্যা প্রিন্ট করার প্রোগ্রাম লেখা হবে। এটি আগের প্রোগ্রামের মতোই হবে, তবে প্রতিটি সংখ্যা প্রিন্ট করার আগে সেটি জোড় কি না, তা পরীক্ষা করা হবে। উল্লেখ্য যে, কোনো সংখ্যাকে 2 দিয়ে ভাগ করলে ভাগশেষ যদি 0 হয়, তাহলে সেটি জোড় সংখ্যা।

```
#include <stdio.h>
int main()
{
    int n;
    n = 1;
    while (n <= 100) {
        if (n % 2 == 0){
            printf("%d\n", n);
        }
        n = n + 1;
    }
    return 0;
}
```

প্রোগ্রামটি চাইলে এভাবেও লেখা যায়—

```
#include <stdio.h>
int main()
{
    int n = 2;
    while (n <= 100) {
        printf("%d\n", n);
        n = n + 2;
    }
    return 0;
}
```

প্রোগ্রাম 5.16

উপরের প্রোগ্রামটিতে n-এর মান 2 থেকে শুরু হয়েছে এবং লুপের ভেতরে প্রতিবার n-এর মান 2 করে বাড়ানো হচ্ছে। তাই প্রোগ্রামটি 2 থেকে শুরু করে প্রতিটি জোড় সংখ্যা প্রিন্ট করবে এবং n-এর মান 100-এর চেয়ে বেশি হলে লুপ থেকে বের হয়ে যাবে।

উদাহরণ ১৩

এখন 1 থেকে 100 পর্যন্ত সব পূর্ণসংখ্যার যোগফল নির্ণয় করার প্রোগ্রাম লেখা হবে। যদিও ধারার সূত্র ব্যবহার করে এক লাইনেই এটি করে ফেলা যায়, কিন্তু এখানে লুপ ব্যবহার করে প্রোগ্রামটি তৈরি করা হবে। শুরুতে ধরা হবে যোগফল শূন্য। তারপর যোগফলের সঙ্গে প্রথমে 1 যোগ করা হবে, তারপর 2 যোগ করা হবে, এভাবে 100 পর্যন্ত সব সংখ্যা ওই যোগফলের সঙ্গে যোগ করা হবে।

```
#include <stdio.h>
int main()
{
    int n, sum;
    sum = 0;
    n = 1;
    while (n <= 100)
    {
        sum = sum + n;
        n = n + 1;
    }
    printf("Result: %d\n", sum);
    return 0;
}
```

প্রোগ্রাম 5.17

do-while loop: এ ক্ষেত্রে শর্ত- এর মান লুপ স্টেটমেন্ট কার্যকর হওয়ার পর নির্ধারিত হয়। অর্থ্যাৎ লুপ শর্ত পূরণ হোক বা না হোক ১ টি বারের জন্য কার্যকর হয়। এ ধরনের লুপকে exit controlled loop বলে।

do-while loop এর সিলট্যাক্স হচ্ছে :

```
do
{
    statement;
}   while (condition );
```

উদাহরণ ১১ প্রোগ্রামটিকে do-while লুপ ব্যবহার করে লেখা যায়।

```
#include <studio.h>
int main ()
{
        int n;
         n=1;
          do
       {
       printf ("%d\n",  n);
       n= n+1;

       } while(n<-100),
return   0;
}
```

প্রোগ্রাম 5.18

নিজে করি ৭ : লুপ ব্যবহার করে 1 থেকে 500 পর্যন্ত সব বেজোড় সংখ্যার যোগফল নির্ণয় করার ফ্লোচার্ট তৈরি করি এবং প্রোগ্রাম লিখি।

for লুপ

সি প্রোগ্রামিং ভাষায় for লুপের সিনট্যাক্স হচ্ছে এরকম—

```
for (initialization; condition; increment)
{
    statement;
    ...
}
```

1 থেকে 100 পর্যন্ত সংখ্যাগুলো যোগ করার প্রোগ্রামটি যদি for লুপ ব্যবহার করে লেখা হয়, সেটি দাঁড়াবে এমন—

```
#include <stdio.h>
int main()
{
    int n, sum;
    sum = 0;
    for(n = 1; n <= 100; n = n + 1) {
        sum = sum + n;
    }
    printf("Result: %d\n", sum);
    return 0;
}
```

প্রোগ্রাম 5.19

নিজে করি ৮ : এখন পর্যন্ত while লুপ ব্যবহার করে বইতে যেসব প্রোগ্রাম তৈরি করা হয়েছে, সেগুলো for লুপ ব্যবহার করে করতে হবে।

Continue স্টেটমেন্ট

Continue স্টেটমেন্ট লুপের ভিতরে ব্যবহৃত হয়। যখন Continue স্টেটমেন্ট এর সাথে ব্যবহৃত শর্তটি সঠিক হয় তখন কন্ট্রোলটি লুপের প্রথমে চলে যায় অন্যথায় পরবর্তী স্টেটমেন্ট কার্যকর হয়।

উদাহরণ:

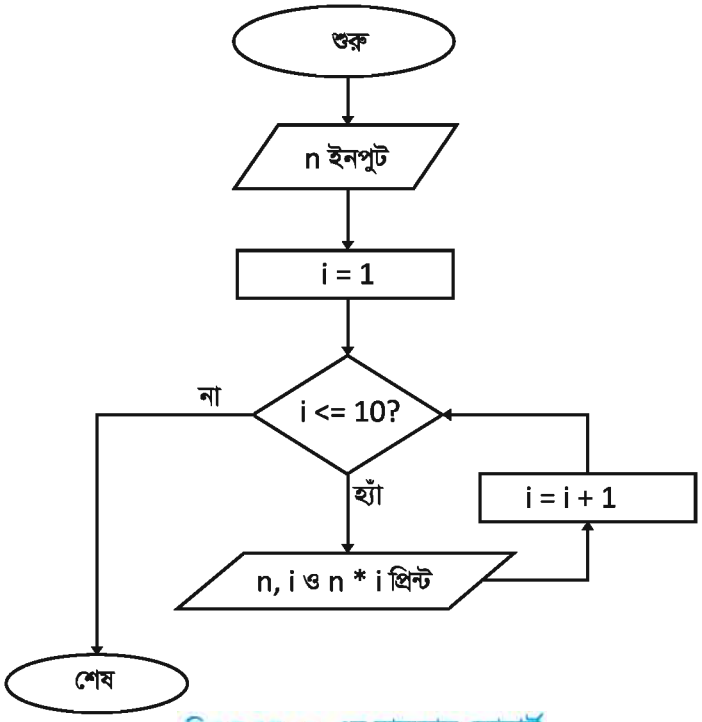
```
#include <stdio.h>
int main ()
{
      int i;
      for (i=1,   i <=5;    i=i+1)
      {
      if {i--}
      continue;
      printf ("%d", i);
      }
return 0;
}
```

প্রোগ্রাম 5.20

ফলাফল হবে: 2345 কারণ যখন i এর মান 1 হবে তখন সংখ্যার মানটি প্রদর্শিত না হয়ে লুপের প্রথমে কন্ট্রোলটি চলে যাবে। ফলে 1 লেখাটি প্রদর্শিত হবে না।

উদাহরণ ১৪

এখন for লুপ ব্যবহার করে নামতা লেখার প্রোগ্রাম তৈরি করতে হবে। প্রথমে ফ্লোচার্ট (চিত্র 5.12) তৈরি করে দেখানো হবে, তারপর প্রোগ্রাম লেখা হবে।



চিত্র 5.10 : n-এর নামতার ফ্লোচার্ট

```
#include <stdio.h>
int main()
{
    int i, n;
    scanf("%d", &n);
    for(i = 1; i <= 10; i = i + 1) {
        printf("%d x %d = %d\n", n, i, n * i);
    }
    return 0;
}
```

প্রয়োজন হলে একটি লুপের ভেতরে আরো লুপ ব্যবহার করা যায়। একে বলে নেস্টেড লুপ (nested loop)।

অ্যারে (Array)

একটি ভ্যারিয়েবলে একই সময়ে কেবল একটি মান রাখা যায়। কিন্তু অনেক সময় একই ধরনের অসংখ্য ভ্যারিয়েবল নিয়ে কাজ করতে হয়। যেমন— একটি ক্লাসের একশজন শিক্ষার্থীর প্রাপ্ত নম্বর আউটপুট দেওয়া।

সি প্রোগ্রামিং ভাষায় এজন্য একটি বিশেষ ডেটা স্ট্রাকচার (data structure) আছে, যার নাম অ্যারে। অ্যারেতে একই ধরনের একাধিক ডেটা রাখা যায়। অ্যারে তৈরি করার সিনট্যাক্স হচ্ছে—

```
data_type name[number of elements];
```

উদাহরণ ১৫

নিচের প্রোগ্রামটিতে একটি অ্যারে তৈরি করা হবে, যেখানে পাঁচজন শিক্ষার্থীর একটি পরীক্ষায় প্রাপ্ত নম্বর রাখা হবে।

```
#include <stdio.h>
int main()
{
    int marks[5];

    // assign marks to array
    marks[0] = 87;
    marks[1] = 82;
    marks[2] = 76;
    marks[3] = 85;
    marks[4] = 88;

    /* now print the marks */
    printf("%d\n", marks[0]);
    printf("%d\n", marks[1]);
    printf("%d\n", marks[2]);
    printf("%d\n", marks[3]);
    printf("%d\n", marks[4]);
    return 0;
}
```

প্রোগ্রাম 5.21

উল্লেখ্য যে, প্রোগ্রামটিতে এক জায়গায় // চিহ্নের পরে, আরেক জায়গায় /* ... */ চিহ্নের ভেতরে কিছু কথা লেখা হয়েছে। এগুলোকে বলা হয় মন্তব্য বা কমেন্ট (comment)। প্রোগ্রাম কম্পাইল ও রান করার সময় এই কমেন্টগুলো কোডের অংশ হিসেবে বিবেচনা করা হয় না। প্রোগ্রামারদের নিজেদের সুবিধার্থে কমেন্ট ব্যবহার করা হয়। তাই কোনো লাইনে // থাকলে সেই লাইনে তার পরের অংশগুলো আর প্রোগ্রামের অংশ বলে ধরা হয় না। একাধিক লাইনজুড়ে কমেন্ট লিখতে চাইলে /* দিয়ে শুরু এবং */ দিয়ে শেষ করতে হয়।

প্রোগ্রাম 5.21-এ marks নামের যেই অ্যারেটি তৈরি করা হয়েছে (Int marks[5];), সেখানে বলে দেওয়া হয়েছে যে, অ্যারেটি ইন্টিজার টাইপের অর্থাৎ অ্যারের সব উপাদান হবে ইন্টিজার আর অ্যারেতে মোট 5টি উপাদান থাকবে। অ্যারের প্রথম উপাদান থাকে 0-তম ঘরে, দ্বিতীয় উপাদান থাকে 1-তম ঘরে, তৃতীয় উপাদান থাকে 2-তম ঘরে, এরকমভাবে n-তম উপাদান থাকে $(n-1)$-তম ঘরে। এই ঘরগুলোকে বলা হয় অ্যারের ইনডেক্স (index)। মনে রাখতে হবে যে, সি প্রোগ্রামিং ভাষায় অ্যারের ইনডেক্স 0 থেকে শুরু হয়, 1 থেকে নয়। তাহলে marks অ্যারেটিতে বিভিন্ন মান থাকবে নিচের চিত্রের মতো,

Value	87	82	76	85	88
Index	0	1	2	3	4

ইনডেক্স থাকার একটি সুবিধা হচ্ছে যে, এখানে লুপ ব্যবহার করা যায়। যেমন— পাঁচবার printf() স্টেটমেন্ট না লিখে এভাবেও লেখা যেত—

```
for (i = 0; i < 5; i = i + 1)
{
    printf("%d\n", marks[i]);
}
```

আবার অ্যারেতে বিভিন্ন মান অ্যাসাইন করার কাজটিও সংক্ষেপে করা যায় এভাবে—

```
int marks[] = {87, 82, 76, 85, 88};
```

এখানে marks-এ বলে দেওয়া নেই কয়টি উপাদান থাকবে, তবে দ্বিতীয় বন্ধনীর ভেতরের উপাদানগুলোর সংখ্যা থেকেই কম্পাইলার বুঝে নেয় যে অ্যারেতে কয়টি উপাদান থাকবে।

আবার ব্যবহারকারীর কাছ থেকে ইনপুট নিতে চাইলে সেটিও সহজে করা যায় এভাবে—

```
for (i = 0; i < 5; i = i + 1)
{
    scanf("%d", &marks[i]);
}
```

মনে রাখতে হবে, অ্যারের ইনডেক্স সব সময় হবে একটি পূর্ণসংখ্যা, যেটি 0 থেকে শুরু হবে। আর অ্যারেতে n সংখ্যক উপাদান থাকলে অ্যারের ইনডেক্স-এর সর্বোচ্চ মান হবে $n - 1$।

উদাহরণ ১৬

একটি অ্যারেতে দশটি সংখ্যা রাখা আছে। সংখ্যাগুলোর যোগফল বের করতে হবে—

```
#include <stdio.h>
int main()
{
    int numbers[10] = {9, 76, 2, 45, 3, 81, 25, 33, 71, 10};
    int i, sum;
    sum = 0;
    for (i = 0; i < 10; i = i + 1) {
        sum = sum + numbers[i];
    }
    printf("Sum: %d\n", sum);
    return 0;
}
```

প্রোগ্রাম 5.22

একটি ফাংশনে যখন কোনো ভ্যারিয়েবল ডিক্লেয়ার করা হয় (যেমন int sum), তখন সেই ভ্যারিয়েবলের ভেতরে কোনো মান দেওয়া থাকে না। সেই ভ্যারিয়েবলটির ভেতরে যে কোনো মান থাকতে পারে, যাকে গারবেজ (garbage) মান বলা হয়। তাই ভ্যারিয়েবলটির ভেতর যদি 0 রাখার প্রয়োজন হয়, তখন এর মধ্যে 0 অ্যাসাইন করতে হবে (যেমন sum = 0)। আর যোগফল নির্ণয়ের প্রোগ্রামটিতে এমনটি করতে হয়েছে কারণ sum = sum + numbers[i] স্টেটমেন্ট চলার আগে sum-এর মান 0 করে দেওয়ার ফলে 0 +9 অর্থাৎ 9 সংখ্যাটি sum-এর মধ্যে আবার রাখা যাচ্ছে।

a = a + b; এই স্টেটমেন্টটি সি ভাষায় আরেকভাবে লেখা যায় : a += b; তাহলে প্রোগ্রাম 5.22-এ for লুপটি এভাবে লেখা যায়,

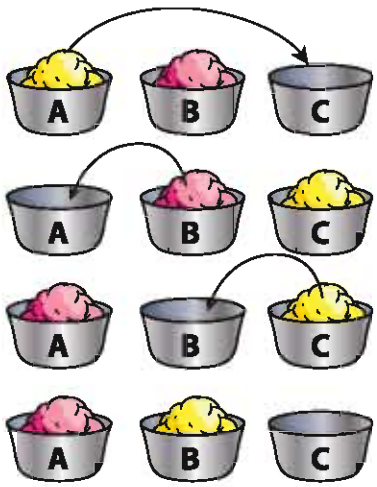
```
for (i = 0; i < 10; i += 1)
{
    sum += numbers[i];
}
```

আবার i += 1 (বা, i = i + 1)-কে i++ লেখা যায়। এটি একটি সংক্ষিপ্ত রূপ। যে কোনো একভাবে লিখলেই চলে।

উদাহরণ ১৭

একটি অ্যারেতে পাঁচটি সংখ্যা আছে। একটি প্রোগ্রাম লিখে সংখ্যাগুলোর ক্রম উল্টে দিতে হবে। অর্থাৎ অ্যারেতে যদি 1, 2, 3, 4, 5 থাকে, তাহলে প্রোগ্রামটি অ্যারেতে 5, 4, 3, 2, 1 নিয়ে আসবে।

প্রোগ্রামটি লেখার আগে একটি অপেক্ষাকৃত সহজ প্রোগ্রাম করে দেখতে হবে। ধরা যাক, দুটি ভ্যারিয়েবল আছে, a ও b। এখন একটি প্রোগ্রাম লিখতে হবে যেন, a-এর মান b-তে চলে আসে আর b-এর মান a-তে চলে আসে। কাজটি করার উপায় কী? একটি সহজ উপায় হচ্ছে, অতিরিক্ত একটি ভ্যারিয়েবল c ব্যবহার করা। তারপরে c-এর ভেতরে a-এর মান অ্যাসাইন করা। তাহলে এখন c ও a-তে একই মান থাকবে। এখন b-এর মান a-তে অ্যাসাইন করা হবে। তাহলে c-তে থাকবে a-এর আসল মান, a ও b-তে থাকবে b-এর মান। তারমানে b-এর মান কিন্তু a-তে চলে এলো। এখন, a-এর আসল মান b-তে আনতে পারলেই কাজ শেষ। c-এর মধ্যে a-এর আসল মান আছে। তাই b = c; লিখলেই কাজ হয়ে যাবে। বিষয়টি অনেকটা নিচের ছবির মতো।



চিত্র 5.11 : ছবিতে a পাত্রে হলুদ রঙের আইসক্রিম এবং b পাত্রে গোলাপি রঙের আইসক্রিম রয়েছে। আমরা চাই, a পাত্রের আইসক্রিম b পাত্রে নিয়ে আসতে এবং b পাত্রের আইসক্রিম a পাত্রে নিয়ে আসতে

```
#include <stdio.h>
int main()
{
    int a = 15, b = 9;
    int c;
```

```
    c = a;
    a = b;
    b = c;
    printf("Value of a is %d, value of b is %d\n", a, b);
    return 0;
}
```

প্রোগ্রাম 5.23

নিজে করি ৯ : উপরের প্রোগ্রামটির অ্যালগরিদম লিখতে হবে এবং ফ্লোচার্ট তৈরি করতে হবে।

এখন আসল সমস্যাটির সমাধান করা হবে। অ্যারের প্রথম উপাদনের সঙ্গে শেষ উপাদানের মানের অদলবদল করা হবে, তারপর অ্যারের দ্বিতীয় উপাদানের সঙ্গে অ্যারের শেষ উপাদানের আগের উপাদানের মানের অদলবদল করা হবে।

প্রোগ্রামটি লেখা যায় এভাবে—

```
#include <stdio.h>

int main()
{
    int ara[] = {10, 20, 30, 40, 50};
    int n = 5,i;
    int temp;

    for (i = 0; i < n / 2; i += 1)
    {
        // exchange value of ara[i] and ara[n-1-i]
        temp = ara[i];
        ara[i] = ara[n-1-i];
        ara[n-1-i] = temp;
    }

    for (i = 0; i < n; i += 1)
    {
        printf("%d\n", ara[i]);
    }

    return 0;
}
```

প্রোগ্রাম 5.24

উপরের প্রোগ্রামটি কম্পাইল ও রান করে দেখতে হবে।

নিজে করি ১০ : প্রথম for লুপে শর্ত ব্যবহার করা হয়েছে i < n / 2, এর বদলে i < n ব্যবহার করলে কী হতো সেটি চিন্তা করে বের করতে হবে।

সি প্রোগ্রামিং ভাষায় ক্যারেক্টার টাইপের ভ্যারিয়েবলে একটি অক্ষর রাখা যায়। যদি একাধিক অক্ষর রাখতে হয়, তখন ক্যারেক্টার টাইপের অ্যারে ব্যবহার করা হয়। একে বলা হয় স্ট্রিং (string)। বিভিন্ন প্রোগ্রামিং ভাষায় স্ট্রিংয়ের জন্য পৃথক ডেটা টাইপ থাকলেও সি-তে আলাদা কোনো ডেটা টাইপ নেই।

উদাহরণ ১৮

নিচের প্রোগ্রামের মাধ্যমে দেখানো হবে সি-তে কীভাবে স্ট্রিং ইনপুট নেওয়া যায় ও আউটপুট দেওয়া যায়,

```
#include <stdio.h>

int main()
{
    char name[80];

    scanf("%s", name);

    printf("%s\n", name);

    return 0;
}
```

প্রোগ্রাম 5.25

যেই স্ট্রিং ইনপুট দেওয়া হবে, প্রোগ্রামটি সেই স্ট্রিং আউটপুট হিসেবে প্রিন্ট করবে। একটি স্ট্রিংয়ের শেষ অক্ষরটি হবে নাল ক্যারেক্টার ('\0')। তাই কোনো স্ট্রিংয়ে যদি বলে দেওয়া হয় সর্বোচ্চ 80টি ঘর থাকবে (name[80]), তাহলে এখানে আসলে সর্বোচ্চ 79টি অক্ষর রাখা যাবে। শেষ ঘরটি নাল ক্যারেক্টারের জন্য বরাদ্দ রাখতে হবে।

সাধারণ ইন্টিজার অ্যারেতে ইনপুট নিতে হলে যেমন একটি লুপ ব্যবহার করে একটি একটি করে সংখ্যা ইনপুট নিতে হয়, ক্যারেক্টার অ্যারে বা স্ট্রিংয়ের ক্ষেত্রে তার প্রয়োজন হয় না। scanf() ফাংশনের ভেতরে %s ব্যবহার করে সম্পূর্ণ স্ট্রিংটি একবারে ইনপুট নেওয়া যায়। তবে স্ট্রিংয়ের ভেতরে কোনো স্পেস (space) ক্যারেক্টার থাকতে পারবে না।

নিচের ছবিতে একটি স্ট্রিং "Bangla" কীভাবে অ্যারেতে থাকে, সেটি দেখানো হয়েছে—

Value	'B'	'a'	'n'	'g'	'l'	'a'	'\0'
Index	0	1	2	3	4	5	6

উদাহরণ ১৯

এখন একটি প্রোগ্রাম লেখা হবে, যেটি একটি স্ট্রিংয়ে কতগুলো অক্ষর বা ক্যারেক্টার আছে, সেটি বের করবে—

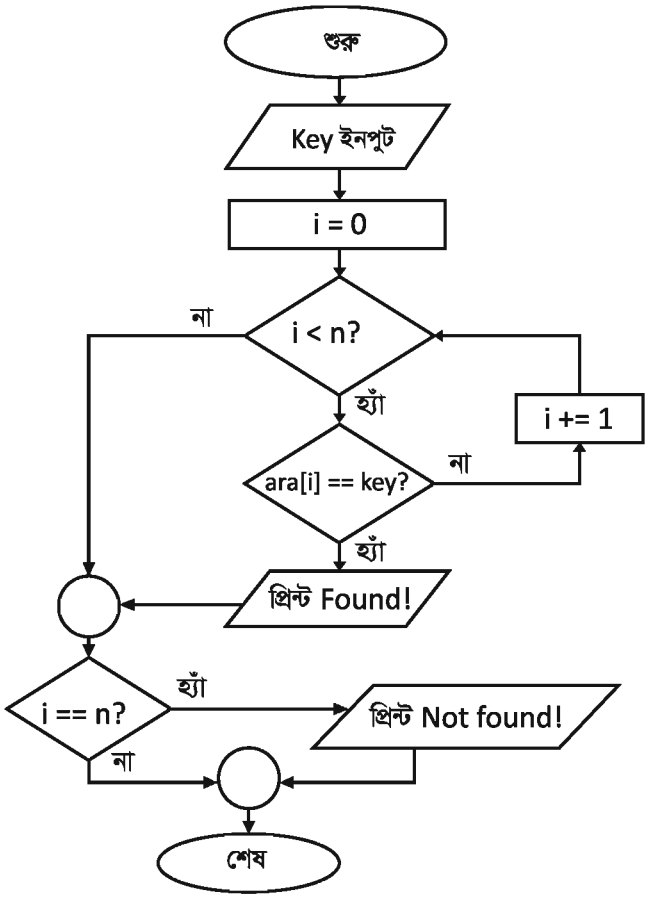
```
#include <stdio.h>
int main()
{
    char name[80];
    int i, length;
    scanf("%s", name);
    i = 0;
    while (name[i] != '\0')
    {
        i = i + 1;
    }

    length = i;
    printf("%s has %d characters.\n", name, length);
    return 0;
}
```

প্রোগ্রাম 5.26

এখানে i-তে 0 অ্যাসাইন করা হয়েছে। তারপর while লুপের ভেতরে শর্ত পরীক্ষা করা হচ্ছে যে, name[i]-এর মান নাল ক্যারেক্টার কি না। যদি না হয়, তাহলে লুপের ভেতরে i-এর মান এক বাড়ানো হয়েছে। যখন name[i]-এর মান নাল ক্যারেক্টারের সমান হবে, তখন প্রোগ্রামটি লুপ থেকে বের হয়ে যাবে। আর i-এর মানই হবে স্ট্রিংয়ের দৈর্ঘ্য, যেটি length নামক ভ্যারিয়েবলে অ্যাসাইন করা হয়েছে। উল্লেখ্য যে, একটি স্ট্রিংয়ে মোট অক্ষরের সংখ্যাকে সেই স্ট্রিংয়ের দৈর্ঘ্য বলা হয়।

উদাহরণ ২০

একটি অ্যারেতে অনেকগুলো সংখ্যা আছে। একটি নির্দিষ্ট সংখ্যা ইনপুট দেওয়া হবে এবং সংখ্যাটি ওই অ্যারেতে আছে কি না, সেটি বের করতে হবে। প্রথমে ফ্লোচার্ট আঁকতে হবে, তারপরে কোড লিখতে হবে।



চিত্র 5.12 : অ্যারেতে সংখ্যা খোঁজার ফ্লোচার্ট

কোড—

```
#include <stdio.h>

int main()
{
    int ara[] = {1, 2, 3, 5, 8, 13, 21, 34, 55};
    int key, i, n;

    n = 9;

    scanf("%d", &key);

    for (i = 0; i < n; i += 1)
    {
        if (ara[i] == key)
        {
            printf("%d is found in the array.\n", key);
            break;
```

```
        }
    }

    if (i == n)
    {
        printf("%d is not found in the array.\n", key);
    }

    return 0;
}
```

প্রোগ্রাম 5.27

উপরের প্রোগ্রামে break স্টেটমেন্ট ব্যবহার করা হয়েছে। এই স্টেটমেন্ট এক্সিকিউট হলে লুপের ভেতর থেকে প্রোগ্রামটি বের হয়ে যাবে। key যদি অ্যারেতে পাওয়া যায়, তাহলে আর খোঁজার কোনো প্রয়োজন নেই, তাই লুপ থেকে বের হয়ে যেতে হবে। লুপ থেকে বের হওয়ার তাহলে দুটি উপায়, এক হচ্ছে break; এক্সিকিউট হওয়া, আর নইলে সব সংখ্যা পরীক্ষা করা হয়ে গেলে i-এর মান n-এর সমান হয়ে যাবে, তখন i < n শর্তটি মিথ্যা হয়ে যাবে আর প্রোগ্রামটি লুপ থেকে বের হয়ে যাবে। তাই for লুপের ব্লকের বাইরে পরীক্ষা করা হচ্ছে যে, i আর n-এর মান সমান কি না। যদি সমান হয়, তাহলে বুঝতে হবে break স্টেটমেন্ট এক্সিকিউট হয়নি, অর্থাৎ সংখ্যাটি খুঁজে পাওয়া যায়নি। এই পদ্ধতিতে বলা হয় লিনিয়ার সার্চ (linear search)।

নিজে করি ১১ : একটি অ্যারেতে ছয়টি সংখ্যা আছে, সেগুলো হচ্ছে যথাক্রমে 5, 8, 1, 9, 4, 10। এখান থেকে 4 সংখ্যাটি লিনিয়ার সার্চ পদ্ধতিতে খুঁজে বের করার ধাপগুলো দেখাই (কোড না লিখে)।

ফাংশন (Function)

প্রোগ্রামাররা বিভিন্ন সময় যখন প্রোগ্রাম লেখে, তখন দেখা যায়, একই কাজ একাধিকবার করতে হচ্ছে। এ কাজগুলো একসঙ্গে একটি ফাংশনের মধ্যে লেখা যায়। তখন কেবল সেই ফাংশনটি কল করলেই চলে, ভেতরের কাজগুলো আবার নতুন করে লিখতে হয় না। এমনকি, ফাংশনটি ভেতরে কীভাবে কাজ করছে, সেটি না জানলেও ফাংশনটি ব্যবহার করতে সমস্যা হয় না।

ইতিমধ্যে বইতে printf() ও scanf() ফাংশনের ব্যবহার দেখানো হয়েছে। এখন স্ক্রিনে প্রিন্ট করার জন্য অনেক কাজ করতে হয় বা কোড লিখতে হয়। সেগুলো printf() ফাংশনের ভেতরে বলা আছে। কিন্তু সি প্রোগ্রামাররা সরাসরি printf() ফাংশন ব্যবহার করে, printf()-এর ভেতরে যে কোড লেখা আছে, সেটি যদি বারবার লিখতে হতো, তাহলে প্রোগ্রামারদের কষ্টও বেড়ে যেত, কোডের আকারও বেড়ে যেত। তেমনি scanf() ফাংশনের বেলাতেও ব্যাপারটি সত্য। ফাংশনটি কীভাবে ব্যবহার করতে হবে, সি প্রোগ্রামারদের এটুকু জানাই যথেষ্ট। এই ফাংশন দুটো ব্যবহার করার জন্য stdio.h নামক হেডার ফাইলটি ইনক্লুড করতে হয়। একটি হেডার ফাইল ইনক্লুড করলে ওই হেডার ফাইলের ভেতরে যেসব ফাংশন তৈরি করে দেওয়া

থাকে, সেগুলো ব্যবহার করা যায়। সি প্রোগ্রামিং ভাষায় এরকম অনেক হেডার ফাইল তৈরি করে দেওয়া আছে। এগুলোকে লাইব্রেরিও বলে। এছাড়া প্রয়োজন হলে প্রোগ্রামার নতুন হেডার ফাইল তৈরি করে নেয়।

একটি ফাংশন ব্যবহার করতে হলে তিনটি জিনিস জানতে হয়। ফাংশনটি কী কাজ করে, ফাংশনের ভেতরে কী কী ডেটা পাঠাতে হবে, আর ফাংশনটি কী ডেটা রিটার্ন করে। যেমন— math.h হেডার ফাইলে একটি ফাংশন আছে যার কাজ হচ্ছে বর্গমূল বের করা। ফাংশনটির প্রোটোটাইপ হচ্ছে - double sqrt(double arg); এখানে প্রথম double হচ্ছে ফাংশনটির রিটার্ন টাইপ, অর্থাৎ ফাংশনটি কী টাইপের ডেটা রিটার্ন করে। ফাংশন যদি কোনো ডেটা রিটার্ন না করে, তখন রিটার্ন টাইপ হয় void। তারপরে sqrt হচ্ছে— ফাংশনের নাম। এরপর প্রথম বন্ধনীর ভেতরে double arg লেখা, যার অর্থ হচ্ছে ফাংশনটি ইনপুট হিসেবে একটি double টাইপের ডেটা গ্রহণ করে— একে বলা হয় ফাংশনের প্যারামিটার (parameter)। আর ফাংশনটি যখন ব্যবহার করা হয়, তখন প্যারামিটারের জায়গায় যে ডেটা পাঠানো হয়, তাকে বলা হয় আর্গুমেন্ট (argument)। নিচের উদাহরণে ফাংশনটির ব্যবহার দেখানো হলো।

উদাহরণ ২১

```
#include <stdio.h>
#include <math.h>

int main()
{
    double num, root;

    scanf("%lf", &num);

    root = sqrt(num);

    printf("Square root of %lf is %lf\n", num, root);

    return 0;
}
```

প্রোগ্রাম 5.28

main() ফাংশনের শেষে কেন return 0; স্টেটমেন্টটি লেখা হয়?
সি ল্যাঙ্গুয়েজে লেখা সব প্রোগ্রাম রান করলে কোডের ভেতরে main() ফাংশন থেকে প্রোগ্রামটি চলা শুরু হয়। main() ফাংশন যদি এভাবে ডিক্লেয়ার করা হয়– int main() তাহলে কম্পাইলার ধরে নেয় যে ফাংশনটি যখন এক্সিকিউশন শেষ হবে তখন সে একটি ইন্টিজার রিটার্ন করবে। তাই ফাংশনের শেষে কোনো একটি ইন্টিজার রিটার্ন করতে হবে। প্রচলিত নিয়মে 0 রিটার্ন করা হয়, প্রোগ্রামটি ঠিকভাবে কোনো সমস্যা ছাড়াই চলেছে সেটা বোঝানোর জন্য। তবে 0-ই যে রিটার্ন করতে হবে এমন কোনো কথা নেই। চাইলে যেকোনো ইন্টিজার-ই রিটার্ন করা যায়।

math.h হেডার ফাইলে আরেকটি ফাংশন হচ্ছে pow (double x, double y);। এই ফাংশনটি প্যারামিটার হিসেবে দুটি double টাইপের সংখ্যা গ্রহণ করে এবং x^y-এর মান হিসেব করে রিটার্ন করে। যেমন— x-এর মান 3 আর y-এর মান 2 পাঠালে ফাংশনটি 9 রিটার্ন করবে।

উদাহরণ ২২

```
#include <stdio.h>
#include <math.h>

int main()
{
    double p, x, y;

    scanf("%lf %lf", &x, &y);

    p = pow(x, y);

    printf("%lf to the power %lf is: %lf\n", x, y, p);

    return 0;
}
```

প্রোগ্রাম 5.29

math.h হেডার ফাইলে এরকম অনেক গাণিতিক ফাংশন তৈরি করে দেওয়া আছে।

একটি স্ট্রিংয়ের দৈর্ঘ্য, অর্থাৎ স্ট্রিংয়ে মোট কয়টি ক্যারেক্টার আছে, সেটি বের করার জন্য ইতিমধ্যে বইতে একটি প্রোগ্রাম লিখে দেখানো হয়েছে। কাজটি চাইলে একটি লাইব্রেরি ফাংশন ব্যবহার করে করা যায়, যার নাম হচ্ছে strlen। ফাংশনটি ইনপুট হিসেবে একটি স্ট্রিং নিবে এবং তার দৈর্ঘ্য রিটার্ন করবে। এই ফাংশনটি রয়েছে string.h হেডার ফাইলে।

উদাহরণ ২৩

```
#include <stdio.h>
#include <string.h>

int main()
{
    char name[80];
    int length;

    scanf("%s", name);

    length = strlen(name);
```

```
    printf("%s has %d characters.\n", name, length);

    return 0;
}
```

প্রোগ্রাম 5.30

দুটি স্ট্রিং সমান নাকি বড়-ছোট, সেটি বের করার জন্য strcmp নামক একটি লাইব্রেরি ফাংশন আছে। যার কাজ হচ্ছে দুটি স্ট্রিং তুলনা করে সমান হলে 0 রিটার্ন করা, প্রথমটি বড় হলে 1 রিটার্ন করা আর প্রথমটি ছোট হলে -1 রিটার্ন করা।

উল্লেখ্য, এখানে বড়-ছোট মানে দৈর্ঘ্যে বড়-ছোট নয়, বরং লেক্সিকোগ্রাফিক্যালি (lexicographically) বড়-ছোট কি না তা বোঝানো হয়েছে। এর মানে হচ্ছে ডিকশনারি বা অভিধানিক ক্রমে সাজালে যে স্ট্রিংটি আগে আসবে তাকে ছোট আর যেটি পরে আসবে তাকে বড় বলে বিবেচনা করা হবে।

উদাহরণ ২৪

```
#include <stdio.h>
#include <string.h>

int main()
{
    char s1[80], s2[80];
    int value;

    scanf("%s %s", s1, s2);

    value = strcmp(s1, s2);

    if (value == 0)
    {
        printf("%s and %s are equal.\n", s1, s2);
    }
    else if (value > 0)
    {
        printf("%s is greater than %s.\n", s1, s2);
    }
    else
    {
        printf("%s is smaller than %s.\n", s1, s2);
    }

    return 0;
}
```

প্রোগ্রাম 5.30

নিজে করি ১৩ : একটি প্রোগ্রাম তৈরি করতে হবে যেটি একটি স্ট্রিং ইনপুট দিলে সেই স্ট্রিংটি প্রিন্ট করবে, তবে quit ইনপুট দিলে প্রোগ্রামটি বন্ধ হয়ে যাবে। যেমন নিচের ছবিতে দেখা যাচ্ছে যে quit ইনপুট দেওয়ার আগ পর্যন্ত যা ইনপুট দেওয়া হচ্ছে, তা-ই প্রিন্ট হচ্ছে।

```
hi
hi
hello
hello
good
good
bad
bad
quit
```

এতক্ষণ বিভিন্ন লাইব্রেরি ফাংশনের ব্যবহার দেখানো হলো। সি প্রোগ্রামিং ভাষায় এরকম শত শত লাইব্রেরি ফাংশন আছে। কম্পাইলারের সঙ্গে দেওয়া ডকুমেন্টেশন কিংবা ইন্টারনেট ঘেঁটে সেই লাইব্রেরি ফাংশনগুলো সম্পর্কে জানা যাবে।

এখন দেখানো হবে কীভাবে নতুন ফাংশন তৈরি করতে হয়।

উদাহরণ ২৫

```c
#include <stdio.h>

float celsius_to_fahrenheit(float celsius);

int main()
{
    float celsius, fahrenheit;

    scanf("%f", &celsius);

    fahrenheit = celsius_to_fahrenheit(celsius);

    printf("Fahrenheit = %f\n", fahrenheit);

    return 0;
}

float celsius_to_fahrenheit(float celsius)
{
    return (celsius * 9 / 5) + 32;
}
```

প্রোগ্রাম 5.31

উপরের প্রোগ্রামে celsius_to_fahrenheit() নামে একটি ফাংশন তৈরি করা হয়েছে। ফাংশনটি একটি সংখ্যা প্যারামিটার হিসেবে নেয় যেটি ডিগ্রি সেলসিয়াস এককে একটি তাপমাত্রা নির্দেশ করে এবং সংখ্যাটি ডিগ্রি ফারেনহাইট এককে রূপান্তর করে রিটার্ন করে। main() ফাংশন লেখার আগে ফাংশনটির প্রোটোটাইপ লেখা হয়েছে—

```
float celsius_to_fahrenheit(float celsius);
```

তারপর main() ফাংশনের পরে ফাংশনটি ইমপ্লিমেন্ট করা হয়েছে। main() ফাংশন থেকে যখন celsius_to_fahrenheit() ফাংশনটি কল (call) করা হচ্ছে, তখন প্রোগ্রাম এই ফাংশনের ভেতরে ঢুকে যাচ্ছে এবং ফাংশন থেকে যখন রিটার্ন করা হচ্ছে, তখন আবার main() ফাংশনের ভেতরে ফেরত আসছে।

নিজে করি ১৩ : ফাংশন ব্যবহার করে যে কোনো সংখ্যা ইনপুট প্রদান করলে ঐ সংখ্যার নামতা প্রদর্শনের জন্যে প্রোগ্রাম লিখ।

উদাহরণ ২৬

1 থেকে 100 পর্যন্ত প্রতিটি সংখ্যার জন্য সংখ্যাটি 3 দ্বারা বিভাজ্য হলে Fizz প্রিন্ট করতে হবে, 5 দ্বারা বিভাজ্য হলে Buzz প্রিন্ট করতে হবে, আর সংখ্যাটি যদি 3 ও 5 উভয় সংখ্যা দ্বারাই বিভাজ্য হয়, তাহলে প্রিন্ট করতে হবে FizzBuzz. প্রতিটি সংখ্যা পরীক্ষা করার কাজটি একটি ফাংশন ব্যবহার করে করতে হবে।

```
#include <stdio.h>

void fizzbuzz(int n);

int main()
{
    int i;

    for (i = 1; i <= 100; i += 1) {
        printf("%d: ", i);
        fizzbuzz(i);
    }

    return 0;
}

void fizzbuzz(int n)
{
```

```
    if (n % 3 == 0 && n % 5 == 0) {
        printf("FizzBuzz\n");
    }
    else if (n % 3 == 0) {
        printf("Fizz\n");
    }
    else if (n % 5 ==0) {
        printf("Buzz\n");
    }
    else {
        printf("\n");
    }
}
```

প্রোগ্রাম 5.32

অনুশীলনী

বহুনির্বাচনি প্রশ্ন

১. সি-ভাষায় সমজাতীয় ডেটা সংরক্ষণের জন্য কোনটি ব্যবহার করা হয়?
ক. ফাংশন খ. পয়েন্টার গ. স্ট্রাকচার ঘ. অ্যারে

২. অ্যালগরিদম ও ফ্লোচার্ট তৈরির পরবর্তী ধাপটা কোনটি?
ক. প্রোগ্রাম পরীক্ষা করা খ. কোড লিখা
গ. সমস্যা সমাধান বর্ণনা ঘ. প্রোগ্রাম রিলিজ করা

৩. সি-ভাষার চলক হলো-
i. student_name
ii. student name
iii. studentname
নিচের কোনটি সঠিক?
ক. i ও ii খ. i ও iii গ. ii ও iii ঘ. i, ii ও iii

নিচের উদ্দীপকটি পড় এবং ৬ ও ৭ নম্বর প্রশ্নের উত্তর দাও :

```
# include<stdio.h>
main(){
        int a = 3, b;
        b = 2*a;
}
```

৪. উদ্দীপকের প্রোগ্রাম রান করলে b এর মান কত হবে?
ক. 3 খ. 4 গ. 5 ঘ. 6

৫. প্রোগ্রাম রান করলে আউটপুট মান 3 হবে যখন-
i. $b = a^{++}$;
ii. $b = a^{--}$;
iii. b+ = a;
নিচের কোনটি সঠিক?
ক. i ও ii খ. i ও iii গ. ii ও iii ঘ. i, ii ও iii

৬. ফরমেট স্পেসিফায়ার হলো-
i. %d
ii/ %if
iii. %c
নিচের কোনটি সঠিক?
ক. i ও ii খ. i ও iii গ. ii ও iii ঘ. i, ii ও iii

৭. 'কম্পাইলার' ও 'ইন্টারপ্রিটার' এর মধ্যে পার্থক্য রয়েছে-
i. প্রোগ্রাম অনুবাদ করার ক্ষেত্রে
ii. মেমোরি স্পেস হ্রাস-বৃদ্ধি
iii. ভুল প্রদর্শনের জন্য
নিচের কোনটি সঠিক?
ক. i ও ii খ. i ও iii গ. ii ও iii ঘ. i, ii ও iii

উদ্দীপকের আলোকে ১২ ও ১৩ প্রশ্নের উত্তর দাও :

```
#include<stdio.h>
main( ){
   int a, s = o;
   for (a = 1; a < = 5; (a= a+1)
   s = s + a;
   printf ("%d", s);
}
```

৮. প্রোগ্রামটির আউটপুট কত?

ক. 0 খ. 1 গ. 5 ঘ. 15

৯. "a" এর মানের কোন কোন পরিবর্তনে আউটপুট 6 হবে?

ক. a = 1, a = a + 2 খ. a = 2, a = a + 1

গ. a = 2, a = a + 2 ঘ. a = 0, a = a + 1

সৃজনশীল প্রশ্ন

১.

ডান পাশের প্রোগ্রামটির জন্য-

ক. সংরক্ষিত শব্দ কী?

খ. K + + ও + + K এর মধ্যকার--- ব্যাখ্যা কর।

গ. উদ্দীপকের প্রোগ্রামটির জন্য একটি প্রবাহচিত্র অঙ্কন কর।

ঘ. উদ্দীপকের প্রোগ্রামটি while লুপ ব্যবহার করে তৈরি করা সম্ভব কি? উত্তরের স্বপক্ষে যুক্তি দাও।

উদ্দীপক

```
#include <stdio.h>
#include <conio.h>
main() {
 int a, s;
 s = 0;
 for (a = 1; a < = 30; a + = 2) {
  s = s + a;
 }
 printf("sum = % d, s);
 getch();
}
```

২.

ডান পাশের প্রোগ্রামটি রান করলে আউটপুট হবে 987654321 অর্থাৎ digits ক্যারেক্টার অ্যারেতে n-এর অংকগুলো বিপরীত ক্রমে এসেছে।

ক. ক্যারেক্টার টাইপের অ্যারেকে প্রোগ্রামিংয়ের ভাষায় কী বলা হয়?

খ. সি প্রোগ্রামিং ভাষায় একটি ইন্টিজার ভ্যারিয়েবলে সর্বোচ্চ কত অঙ্কের সংখ্যা রাখা যায়? ব্যাখ্যা কর।

গ. উদ্দীপকের প্রোগ্রামটির ফ্লোচার্ট তৈরি কর।

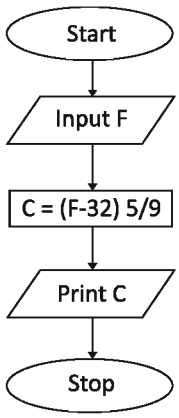
ঘ. প্রোগ্রামটি কীভাবে পরিবর্তন করলে n-এর অংকগুলো সঠিক ক্রমে আসবে যুক্তিসহ প্রমাণ কর।

উদ্দীপক

```
#include <stdio.h>
int main() {
 int i, d;
 int n = 123456789;
 char digits[10];
 i = 0;
 while (n) {
  d = n % 10;
  n = n / 10;
  digits[i] = d + '0';
  i += 1;
 }
 printf("%s\n", digits);
 return 0;
}
```

৩.

উদ্দীপক



ক. কম্পাইলার কী?
খ. অ্যালগরিদম কোডিং এর পূর্বশর্ত- ব্যাখ্যা কর।
গ. উদ্দীপকের সমস্যাটির 'সি' ভাষায় একটি প্রোগ্রাম তৈরি কর।
ঘ. উদ্দীপকটি প্রোগ্রাম তৈরির একটি ধাপ - বিশ্লেষণ কর।

৪. বার্ষিক ক্রীড়া প্রতিযোগিতায় একাদশ শ্রেণির শিক্ষার্থীদের A, B ও C দলে বিভক্ত করা হয়। রোল নম্বর 1 থেকে 30 পর্যন্ত A দলে, 31 থেকে 60 পর্যন্ত B দলে, এবং 61 থেকে 100 পর্যন্ত C দলে অন্তর্ভুক্ত হবে।
ক. প্রোগ্রাম কী?
খ. 'সি' একটি কেস-সেনসিটিভ ভাষা – ব্যাখ্যা কর।
গ. উদ্দীপকে উল্লিখিত দল গঠনের জন্য অ্যালগরিদম লিখ।
ঘ. সি-ভাষায় কন্ডিশনাল স্টেটমেন্ট ব্যবহার করে দল গঠনের জন্য একটি প্রোগ্রাম লিখে এর যৌক্তিকতা বিশ্লেষণ কর।

৫. মাইশা দেখল, তার মামার পচণ্ড জ্বর। সে থার্মোমিটারে মেপে দেখল 103^0F, কিন্তু রুমের তাপমাত্রা 30^0C।
ক. ডেটা টাইপ কী?
খ. কম্পাইলারের তুলনায় ইন্টারপ্রিটার কোন ক্ষেত্রে ভালো-, ব্যাখ্যা কর।
গ. উদ্দীপকে উল্লেখিত থার্মোমিটারের তাপমাত্রাকে সেলসিয়াসে রূপান্তরের জন্য সি ভাষায় প্রোগ্রাম লিখ।
ঘ. উদ্দীপকে উল্লেখিত ফারেনহাইট তাপমাত্রাকে সেলসিয়াসে রূপান্তরের জন্য এলগরিদম নয় ফ্লোচার্টই উত্তম- ব্যাখ্যা কর।

৬. আদনান জামি দুটি সংখ্যা L, S (L<S) এর গ.সা.গু নির্ণয়ের জন্য 'সি' ভাষা প্রোগ্রাম করতে চাচ্ছে। কিন্তু সে প্রোগ্রামটি লজিক কিছুই বুঝতে পারছে না। অবশেষে সে তার আইসিটি শিক্ষকের স্মরণাপন্ন হলেন। তার শিক্ষক তাকে সমস্যাটি কয়েকটি ধাপে ভেঙে প্রত্যেকটি ধাপের চিত্রসহকারে উপস্থাপন করে তাকে বুঝিয়ে দিলেন। এখন আদনান জামির আর কোনো সমস্যা রইল না।
ক. প্রোগ্রামিং কী?
খ. প্রোগ্রামারগণ কোনো বড় প্রোগ্রামকে ছোট ছোট ভাগে ভাগ করে কি সুবিধা পান? বুঝিয়ে বল।
গ. শিক্ষক হিসেবে তুমি সমস্যাটির সমাধান দাও।
ঘ. L = 8 এবং S = 3 হলে উক্ত ধাপগুলো কীভাবে কাজ করবে পর্যায়ক্রমে দেখাও।