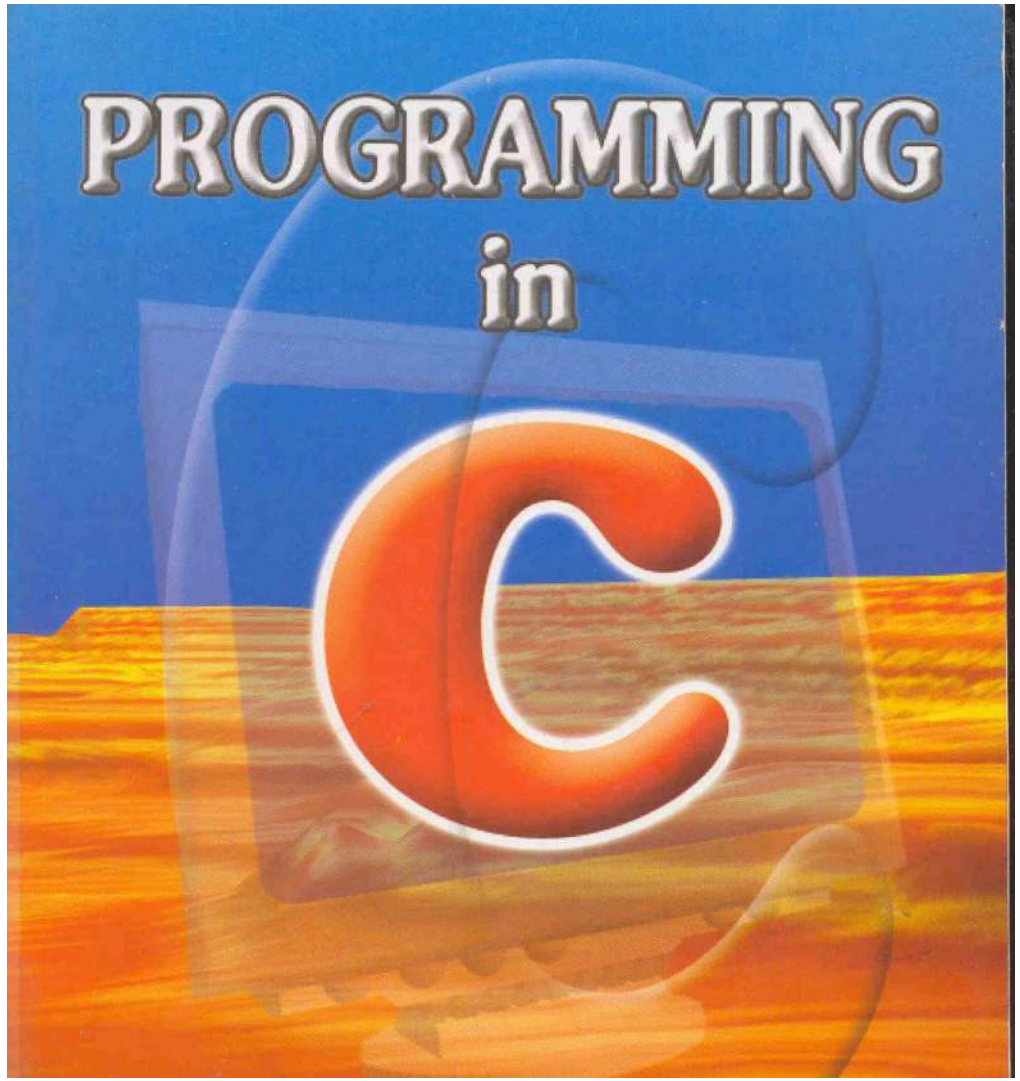

PROGRAMMING
in
C

Chapter-1
Fundamentals of C programming

After completing this chapter we will learn about-

- What is Program & Programming Language
- Generation of programming language
- Compiler, interpreter
- Steps of program planning
- Algorithm & Flowchart
- Some examples of Flowchart & Algorithm

প্রোগ্রামিং কী?

প্রোগ্রামিং ভাষা ব্যবহার করে কোন যন্ত্রকে নির্দেশনা দেওয়াকে বলা হয় প্রোগ্রামিং। অন্যভাবে বলা যায়, প্রোগ্রাম রচনার পদ্ধতি বা কৌশলকে প্রোগ্রামিং বলা হয়।

প্রোগ্রাম কী?

যন্ত্রের মাধ্যমে কোন সমস্যা সমাধানের লক্ষ্যে প্রোগ্রামিং ভাষা ব্যবহার করে প্রোগ্রামারের দেওয়া ধারাবাহিক নির্দেশনার সমষ্টিকে প্রোগ্রাম বলা হয়।

কম্পিউটার আবিষ্কারের শুরুর দিকে প্রোগ্রামের কোন ধারণা ছিলোনা। তখন কোন সমস্যা সমাধান করার জন্য প্রয়োজনীয় নির্দেশসমূহ ধারাবাহিকভাবে ০ ও ১ ইনপুটের মাধ্যমে সমাধান করা হতো।

প্রোগ্রামিং ভাষা কী?

যে ভাষার সাহায্যে একটি যন্ত্রকে নির্দেশনা দিয়ে কোন সমস্যা সমাধান করা যায় তাকে প্রোগ্রামিং ভাষা বলে।

সহজে বলা যায়, প্রোগ্রাম লিখতে বা নির্দেশাবলী সেট করতে যেসকল ভাষা ব্যবহৃত হয় তাদের "প্রোগ্রামিং ভাষা" বলা হয়।

অন্যভাবে বলা যায়, যন্ত্রের মাধ্যমে কোন সমস্যা সমাধানের জন্য ব্যবহৃত শব্দ, বর্ণ, অংক, চিহ্ন প্রভৃতির সমন্বয়ে গঠিত রীতিনীতিকে প্রোগ্রামিং ভাষা বলা হয়।

বিভিন্ন প্রজন্মের প্রোগ্রামিং ভাষা

1945 থেকে শুরু করে এ পর্যন্ত যত প্রোগ্রামিং ভাষা আবিষ্কৃত হয়েছে তাদেরকে বৈশিষ্ট্য অনুযায়ী নিম্নোক্ত পাঁচটি প্রজন্মে ভাগ করা হয়েছে।

১। প্রথম প্রজন্ম – First Generation(1945-1950)

Machine Language (যান্ত্রিক ভাষা)

২। দ্বিতীয় প্রজন্ম – Second Generation(1950-1960)

Assembly Language (অ্যাসেম্বলি ভাষা)

৩। তৃতীয় প্রজন্ম –Third Generation(1960-1970)

High Level Language (উচ্চস্তরের ভাষা)

৪। চতুর্থ প্রজন্ম – Fourth Generation(1970-1980)

Very High Level Language (অতি উচ্চস্তরের ভাষা)

৫। পঞ্চম প্রজন্ম – Fifth Generation(1980-present)

Natural Language (স্বাভাবিক ভাষা)

বিভিন্ন স্তরের প্রোগ্রামিং ভাষা

প্রোগ্রাম রচনার বৈশিষ্ট্যের ভিত্তিতে প্রোগ্রামিং ভাষাসমূহকে আবার বিভিন্ন স্তরে বিভক্ত করা হয়।

১। নিম্নস্তরের ভাষা (Low Level Language)

-Machine Language, Assembly Language

২। মধ্যমস্তরের ভাষা (Mid Level Language)

-C, C++, JAVA, Forth, Dbase, WordStar ইত্যাদি।

৩। উচ্চস্তরের ভাষা (High Level Language- **3GL**)

-Fortran, Basic, Pascal, Cobol, C, C++, C#, Visual Basic, Java, Python ইত্যাদি।

৪। অতি উচ্চস্তরের ভাষা (Very High Level Language- **4GL**)

-Perl, Python, Ruby, SQL, MatLab(MatrixLaboratory) ইত্যাদি।

প্রথম প্রজন্মের ভাষা(1st Generation Language-1GL):

মেশিন ভাষা বা যান্ত্রিক ভাষা (Machine Language):

যে ভাষায় শুধুমাত্র ০ এবং ১ ব্যবহার করে প্রোগ্রাম লেখা হয় তাকে মেশিন বা যান্ত্রিক ভাষা বলে। কম্পিউটারের নিজস্ব ভাষা বা মৌলিক ভাষা হচ্ছে মেশিন ভাষা। এই ভাষায় শুধু মাত্র ০ এবং ১ ব্যবহার করা হয় বলে এই ভাষায় দেওয়া কোনো নির্দেশ কম্পিউটার সরাসরি বুঝতে পারে। ফলে এর সাহায্যে কম্পিউটারের সাথে সরাসরি যোগাযোগ করা যায়।

এটি প্রথম প্রজন্মের এবং নিম্নস্তরের ভাষা। হার্ডওয়্যারের সাথে সরাসরি সম্পর্কিত এবং যন্ত্র নির্ভর বলে এই ভাষাকে নিম্নস্তরের ভাষা বলা হয়। মেশিন ভাষায় লেখা প্রোগ্রামকে অবজেক্ট বা বস্তু প্রোগ্রাম বলা হয়।

দ্বিতীয় প্রজন্মের ভাষা (2nd Generation Language-2GL):

অ্যাসেম্বলি ভাষা (Assembly Language):

যে ভাষায় বিভিন্ন সংকেত বা নেমোনিক ব্যবহার করে প্রোগ্রাম লেখা হয় তাকে অ্যাসেম্বলি ভাষা বলে। অ্যাসেম্বলি ভাষায় প্রোগ্রাম লেখার জন্য ০ ও ১ ব্যবহার না করে বিভিন্ন সংকেত ব্যবহার করা হয়।

এই সংকেতকে বলে সাংকেতিক কোড (Symbolic Code) বা নেমোনিক (mnemonic) এবং এটি সর্বোচ্চ পাঁচটি লেটারের সমন্বয়ে হয়, যেমন- SUB(বিয়োগের জন্য), MUL(গুণের জন্য), ADD(যোগের জন্য), DIV(ভাগের জন্য) ইত্যাদি। এই বৈশিষ্ট্যের জন্য এই ভাষাকে সাংকেতিক ভাষাও বলা হয়।

অ্যাসেম্বলি ভাষা দ্বিতীয় প্রজন্মের এবং নিম্নস্তরের ভাষা। দ্বিতীয় প্রজন্মের কম্পিউটারে এই ভাষা ব্যাপকভাবে প্রচলিত ছিল। এই ভাষা মেশিন ভাষা থেকে উন্নত হলেও উচ্চতর ভাষার সমকক্ষ নয় এবং যন্ত্র নির্ভর হওয়ায় অ্যাসেম্বলি ভাষাকে নিম্নস্তরের ভাষা বলা হয়।

এই ভাষায় লেখা প্রোগ্রাম অনুবাদের প্রয়োজন হয় এবং অনুবাদক প্রোগ্রাম হিসেবে অ্যাসেম্বলার ব্যবহৃত হয়।

তৃতীয় প্রজন্মের ভাষা (3rd Generation Language-3GL):

উচ্চস্তরের ভাষা (High Level Language):

উচ্চস্তরের ভাষা হলো সেই সকল ভাষা যা মানুষের বোধগম্য এবং মানুষের ভাষার কাছাকাছি। যেমন- উচ্চস্তরের ভাষা ইংরেজি ভাষার সাথে মিল আছে এবং এই প্রোগ্রামিং ভাষা যন্ত্র নির্ভর নয়, এই জন্য এসব ভাষাকে উচ্চস্তরের ভাষা বলা হয়।

এটি মানুষের জন্য বুঝা খুব সহজ কিন্তু কম্পিউটার সরাসরি বুঝতে পারে না বলে অনুবাদক প্রোগ্রামের সাহায্যে একে মেশিন ভাষায় রুপান্তর করে নিতে হয়। এটি তৃতীয় প্রজন্মের ভাষা।

যেমন- Fortran, Basic, Pascal, Cobol, C, C++, Visual Basic, Java, Python ইত্যাদি।

উচ্চস্তরের(C) ভাষায় দুটি সংখ্যা যোগ করার প্রোগ্রাম-

```
#include<stdio.h>
main()
{
   int a, b, c;
   scanf(“%d %d”,&a,&b);
   c = a+b;
   printf("%d",c);
 }
```

অনুবাদক প্রোগ্রাম কী?

কম্পিউটার সহ যেকোন মেশিন শুধুমাত্র ০ এবং ১ বুঝতে পারে। অর্থাৎ শুধুমাত্র মেশিন ভাষায় লেখা নির্দেশনা বা প্রোগ্রাম সরাসরি বুঝতে পারে কিন্তু অন্য প্রোগ্রামিং ভাষায় লেখা নির্দেশনা বা প্রোগ্রাম মেশিন সরাসরি বুঝতে পারে না।

আবার বর্তমানের প্রোগ্রামাররা উচ্চস্তরের বিভিন্ন প্রোগ্রামিং ভাষা ব্যবহার করে প্রোগ্রাম লিখে বা মেশিনকে নির্দেশনা দেয়। এজন্য উচ্চস্তরের প্রোগ্রামিং ভাষায় লেখা প্রোগ্রামকে মেশিনের বোধগম্য করতে অনুবাদ করার প্রয়োজন হয়।

যে প্রোগ্রাম বা সফটওয়্যার এই অনুবাদের কাজটি করে থাকে তাকে অনুবাদক প্রোগ্রাম বা অনুবাদক সফটওয়্যার বলে।অনুবাদক সফটওয়্যার হলো এক ধরণের সিস্টেম সফটওয়্যার এবং এটিকে ল্যাংগুয়েজ প্রসেসরও বলা হয়। প্রতিটি প্রোগ্রামিং ভাষার জন্য পৃথক অনুবাদক প্রোগ্রাম রয়েছে।

যে প্রোগ্রাম উৎস(Source) প্রোগ্রামকে বস্তু(Object) প্রোগ্রামে রূপান্তর করে তাকে অনুবাদক প্রোগ্রাম বলে।

যে প্রোগ্রাম বা সফটওয়্যার এই অনুবাদের কাজটি করে থাকে তাকে অনুবাদক প্রোগ্রাম বা অনুবাদক সফটওয়্যার বলে।অনুবাদক সফটওয়্যার হলো এক ধরণের সিস্টেম সফটওয়্যার এবং এটিকে ল্যাংগুয়েজ প্রসেসরও বলা হয়। প্রতিটি প্রোগ্রামিং ভাষার জন্য পৃথক অনুবাদক প্রোগ্রাম রয়েছে।

যে প্রোগ্রাম উৎস(Source) প্রোগ্রামকে বস্তু(Object) প্রোগ্রামে রূপান্তর করে তাকে অনুবাদক প্রোগ্রাম বলে।

মেশিন ভাষায় লেখা প্রোগ্রামকে বলা হয় বস্তু প্রোগ্রাম (Object Program) এবং অন্য যেকোনো ভাষায় লেখা প্রোগ্রামকে বলা হয় উৎস প্রোগ্রাম (Source program)।

অনুবাদক প্রোগ্রাম উৎস প্রোগ্রামকে ইনপুট হিসেবে নেয় এবং বস্তু প্রোগ্রামকে আউটপুট হিসেবে দেয়। প্রোগ্রাম অনুবাদের সময় উৎস প্রোগ্রামে যদি কোন ভুল থাকে, তবে তা সংশোধন করার জন্য ব্যবহারকারীকে Error Message দেয়।

অনুবাদক প্রোগ্রামের প্রকারভেদ -

১। কম্পাইলার (Compiler)

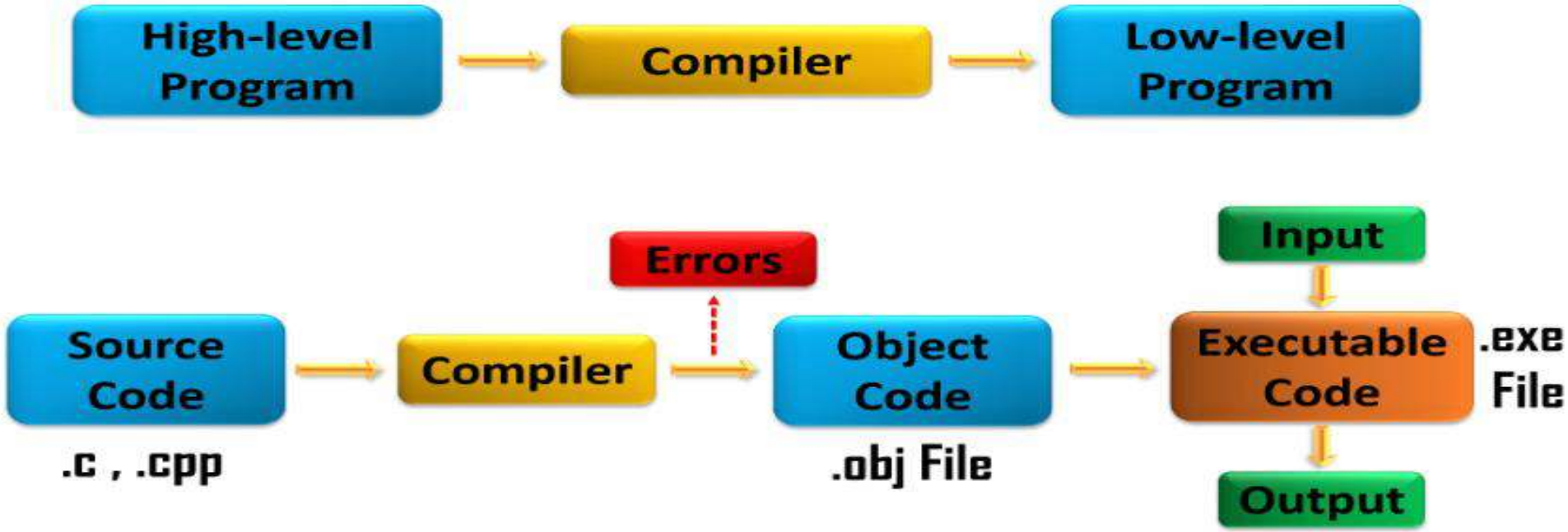
২। ইন্টারপ্রেটার (Interpreter)

৩। অ্যাসেম্বলার (Assembler)

কম্পাইলার কী?

কম্পাইলার হলো এক ধরনের অনুবাদক প্রোগ্রাম যা উচ্চস্তরের ভাষায় লেখা একটি সম্পূর্ণ প্রোগ্রামকে একসাথে পড়ে এবং একসাথে মেশিন বা যান্ত্রিক ভাষায় রূপান্তর করে। অর্থাৎ উৎস প্রোগ্রামকে বস্তু প্রোগ্রামে রূপান্তর করে।

দ্বিতীয় ধাপে ইনপুট উপাত্ত বা ডেটার ভিত্তিতে ফলাফল প্রদর্শনের জন্য বস্তু প্রোগ্রামকে নির্বাহ করে।



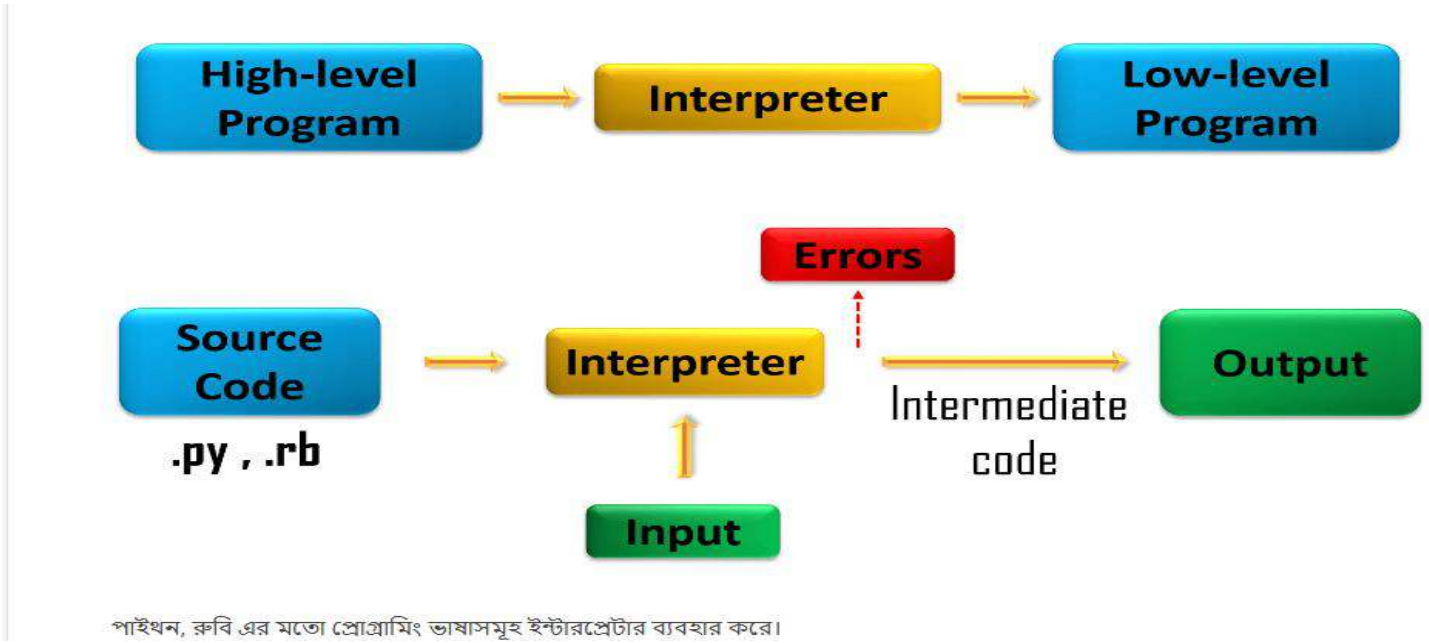
সি, সি ++ এর মতো প্রোগ্রামিং ভাষাসমূহ কম্পাইলার ব্যবহার করে।

ইন্টারপ্রেটার কী?

ইন্টারপ্রেটারও কম্পাইলারের মতো এক ধরনের অনুবাদক প্রোগ্রাম যা উচ্চস্তরের ভাষায় লেখা প্রোগ্রামের একটি করে লাইন পড়ে এবং মেশিন বা যান্ত্রিক ভাষায় রূপান্তর করে।

এক্ষেত্রে উচ্চস্তরের ভাষায় লেখা প্রোগ্রামের একটি করে লাইন পড়ে ও ইন্টারমিডিয়েট কোডে রূপান্তর করে এবং প্রয়োজনীয় ডেটা ইনপুট নিয়ে নির্বাহ করে তাৎক্ষণিক ফলাফল প্রদর্শন করে।

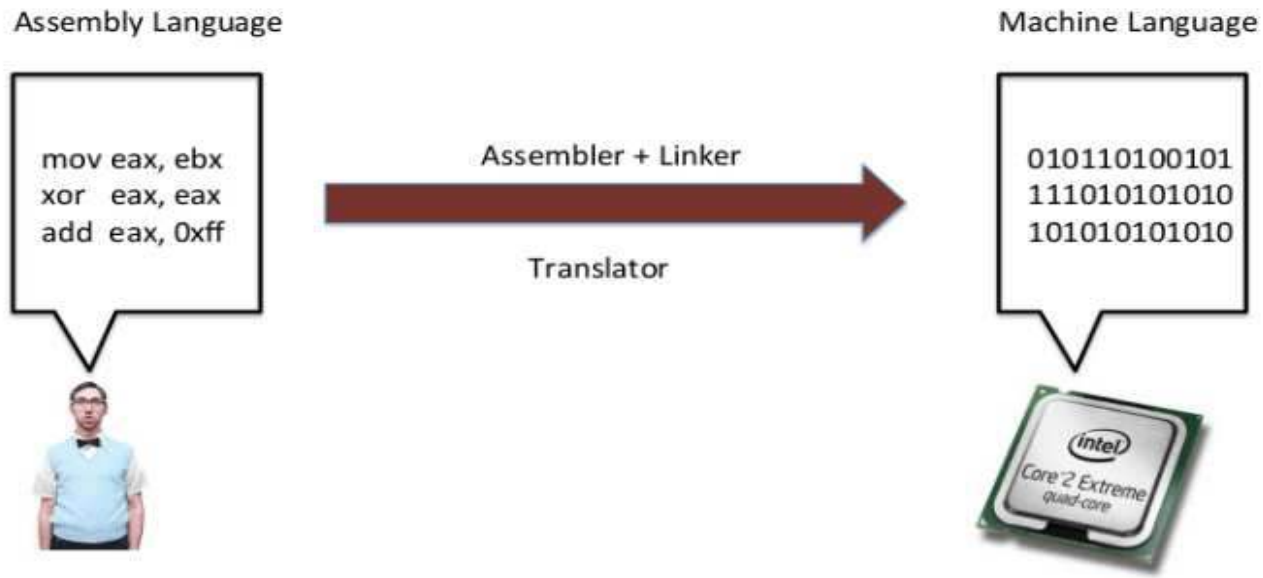
কম্পাইলারের সাথে পার্থক্য হল, কম্পাইলার সম্পূর্ণ উৎস প্রোগ্রামকে একসাথে অবজেক্ট প্রোগ্রামে রূপান্তর করে এবং সর্বশেষ ফলাফল প্রদান করে কিন্তু ইন্টারপ্রেটার উৎস প্রোগ্রামটিকে লাইন-বাই-লাইন রূপান্তর করে ও নির্বাহ করে এবং তাৎক্ষণিক ফলাফল প্রদর্শন করে।



পাইথন, রুবি এর মতো প্রোগ্রামিং ভাষাসমূহ ইন্টারপ্রেটার ব্যবহার করে।

অ্যাসেম্বলার কী?

অ্যাসেম্বলার হলো এক ধরনের অনুবাদক প্রোগ্রাম যা অ্যাসেম্বলি ভাষায় লেখা প্রোগ্রামকে মেশিন ভাষায় রূপান্তর করে। এটি অ্যাসেম্বলি ভাষায় লেখা প্রোগ্রাম বা নেমোনিক কোডকে যান্ত্রিক ভাষায় রূপান্তর করে। এক্ষেত্রে প্রোগ্রামে কোনো ভুল থাকলে Error Message দেয়।



কম্পাইলার ও ইন্টারপ্রেটারের মধ্যে পার্থক্যঃ

কম্পাইলার	ইন্টারপ্রেটার
কম্পাইলার একটি অনুবাদক প্রোগ্রাম যা উচ্চস্তরের প্রোগ্রামিং ভাষায় লেখা একটি সম্পূর্ণ প্রোগ্রামকে একসাথে অনুবাদ করে।	ইন্টারপ্রেটারও এক ধরণের অনুবাদক প্রোগ্রাম যা উচ্চস্তরের প্রোগ্রামিং ভাষায় লেখা একটি প্রোগ্রামকে লাইন বাই লাইন অনুবাদ করে।
ফলে প্রোগ্রাম নির্বাহের জন্য কম সময় প্রয়োজন।	ফলে প্রোগ্রাম নির্বাহের জন্য বেশি সময় প্রয়োজন।
কম্পাইলার দ্বারা একটি প্রোগ্রাম একবার অনুবাদ করা হলে প্রতিবার কাজের পূর্বে পুনরায় অনুবাদ করার প্রয়োজন হয় না।	ইন্টারপ্রেটার দ্বারা একটি প্রোগ্রাম অনুবাদ করা হলে প্রত্যেকবার প্রোগ্রাম নির্বাহের সময় অনুবাদ করার প্রয়োজন হয়।
কম্পাইলার দ্বারা একটি প্রোগ্রাম অনুবাদ করলে পূর্ণাঙ্গ যান্ত্রিক প্রোগ্রামে রূপান্তরিত হয়।	ইন্টারপ্রেটার দ্বারা একটি প্রোগ্রাম অনুবাদ করলে পূর্ণাঙ্গ যান্ত্রিক প্রোগ্রামে রূপান্তরিত হয় না।
ডিবাগিং এবং টেস্টিং এর ক্ষেত্রে ধীর গতি সম্পন্ন।	ডিবাগিং ও টেস্টিং এর ক্ষেত্রে দ্রুত গতি সম্পন্ন।

প্রোগ্রাম তৈরির ধাপসমূহ

একটি প্রোগ্রাম তৈরির মাধ্যমে সাধারণত একটি নির্দিস্ট সমস্যার সমাধান করা হয়ে থাকে। তাই একটি প্রোগ্রাম তৈরি করার জন্য কতগুলো ধাপ অনুসরণ করলে সমস্যাটি সহজে সমাধান করা যায়। ধাপগুলো নিমোক্ত আলোচনা করা হল-

১। সমস্যা নির্দিষ্টকরণ

২। সমস্যা বিশ্লেষণ

৩। প্রোগ্রাম ডিজাইন

৪। প্রোগ্রাম উন্নয়ন

৫। প্রোগ্রাম বাস্তবায়ন

৬। ডকুমেন্টশন

৭। প্রোগ্রাম রক্ষণাবেক্ষণ

সমস্যা নির্দিষ্টকরণঃ

একটি প্রোগ্রাম তৈরির মূল লক্ষ হল কোন একটি সমস্যার সমাধান করা। তাই প্রোগ্রাম তৈরির মাধ্যমে কোন সমস্যা সমাধানের পূর্বে সমস্যাটি অবশ্যই ভালোভাবে চিহ্নিত করতে হবে। সমস্যা নির্দিষ্টকরণের ক্ষেত্রে সমস্যাটি কী, সমস্যার বিষয়বস্তু কী ইত্যাদি বিষয়ে জানতে হবে। ভালভাবে সমস্যা নির্দিষ্ট করতে না পারলে প্রোগ্রাম যতই উন্নত হোক না কেন কাংখিত সমাধান পাওয়া সম্ভব না।

সমস্যা বিশ্লেষণঃ

সমস্যা নির্দিষ্ট করার পরের ধাপটি হল সমস্যা বিশ্লেষণ। সমস্যাটি কীভাবে সমাধান করা যায়, কতভাবে সমাধান করা যায়, যদি একাধিক ভাবে সমাধান করা যায় তাহলে কোনটি সবচেয়ে ইফেক্টিভ সমাধান তা বিশ্লেষণ করাই হল সমস্যা বিশ্লেষণ। এক্ষেত্রে সমস্যাটিকে ছোট ছোট অংশে ভাগ করে সমাধান করার চেষ্টা করা হয়। সমস্যা সমাধানের ক্ষেত্রে কতকগুলো বিষয় এই ধাপে বিবেচনা করা হয়ে থাকে। বিষয়গুলো হল-

১। কোন বিষয়গুলো প্রোগ্রাম ডেভেলপমেন্টের জন্য প্রয়োজন।

২। কোন পদ্ধতিতে প্রোগ্রাম ডিজাইন করা হবে।

ডকুমেন্টেশনঃ

প্রোগ্রাম ডেভেলপমেন্টের সময় ভবিষ্যতে প্রোগ্রাম রক্ষণাবেক্ষণের কথা ভেবে প্রোগ্রামের বিভিন্ন অংশের বিবরণ কমেন্ট হিসেবে লিখে রাখতে হয়। প্রোগ্রামের বিভিন্ন অংশের বিবরণ কমেন্ট হিসেবে লিপিবদ্ধ করাকে প্রোগ্রাম ডকুমেন্টেশন বলে। প্রোগ্রামের ডকুমেন্টেশন লেখা থাকলে যেকোন প্রোগ্রামার খুব সহজেই প্রোগ্রাম আপডেট করতে পারে। প্রোগ্রাম রক্ষণাবেক্ষণে ডকুমেন্টেশনের গুরুত্ব অপরিসীম।

ডকুমেন্টেশনে নিম্নলিখিত বিষয়সমূহ অন্তর্ভুক্ত করা হয়ঃ

১। প্রোগ্রামের বর্ণনা।

২। অ্যালগোরিদম বা ফ্লোচার্ট

৩। নির্বাহের জন্য প্রয়োজনীয় কাজের তালিকা

৪। প্রোগ্রামের আউটপুট

প্রোগ্রাম ডকুমেন্টেশন এর কয়েকটি সুবিধা -

১। একটি প্রোগ্রামের সমস্ত অংশের উপর নজর রাখে

২। প্রোগ্রাম রক্ষণাবেক্ষণ সহজ

৩। ডেভেলোপার ছাড়া অন্য প্রোগ্রামাররা প্রোগ্রামের সমস্ত দিক বুঝতে পারে

৪। প্রোগ্রামের সামগ্রিক মানের উন্নতি করে

প্রোগ্রাম রক্ষণাবেক্ষণঃ

সময়ের সাথে পরিবেশ-পরিস্থিতি পরিবর্তনের কারণে প্রোগ্রামের পরিবর্তন বা আধুনিকীকরণ করা প্রয়োজন হয়। নিম্নোক্ত আউটকাম অর্জনের জন্য প্রোগ্রাম রক্ষণাবেক্ষণ করার প্রয়োজন হয়-

প্রোগ্রাম উন্নয়নঃ

সমস্যা সমাধানের জন্য যে অ্যালগরিদম বা ফ্লোচার্ট তৈরি করা হয়েছে তা কোনো একটি উচ্চস্তরের প্রোগ্রামিং ভাষা ব্যবহার করে প্রোগ্রামে রূপদান করাকে বলা হয় প্রোগ্রাম উন্নয়ন। এক্ষেত্রে সমস্যার ধরণ অনুযায়ী উচ্চস্তরের প্রোগ্রামিং ভাষা হিসেবে C, C++, java, python ইত্যাদি ব্যবহৃত হয়।

প্রোগ্রাম বাস্তবায়নঃ

প্রোগ্রাম বাস্তবায়ন ধাপে প্রোগ্রাম এর টেস্টিং এবং ডিবাগিং করা হয়ে থাকে।

১। টেস্টিং

২। ডিবাগিং

টেস্টিংঃ প্রোগ্রাম টেস্টিং হচ্ছে, কোনো প্রোগ্রাম উন্নয়ন বা কোডিং সম্পন্ন করার পর প্রোগ্রামটি রান করছে কিনা বা যে ধরনের আউটপুট বা ফলাফল হওয়া উচিৎ তা ঠিকমতো আসছে কিনা তা যাচাই করা। এই ধাপে ভিন্ন ভিন্ন ইনপুট দিয়ে আউটপুটের অবস্থা পর্যবেক্ষণ করা হয়। এক্ষেত্রে যদি কোন অসঙ্গতি পাওয়া যায় তবে বুঝতে হবে প্রোগ্রাম কোডিংয়ের কোথাও ভুল হয়েছে। প্রোগ্রামে সাধারণত নিচের ভুলগুলো পরিলক্ষিত হয়। যথা:

১। ব্যাকরণগত ভুল (Syntax Error)

২। যৌক্তিক ভুল (Logical Error)

৩। রান টাইম বা এক্সিকিউশন টাইম ভুল (Run Time or Execution Time Error)

সিনট্যাক্স ভুল/ব্যাকরণগত ভুলঃ প্রোগ্রামের মধ্যে প্রোগ্রামিং ভাষার ব্যাকরণগত যেসব ভুল থাকে তাকে বলা হয় সিনট্যাক্স ভুল। যেমন- বানান ভুল,কমা, ব্র্যাকেট ঠিকমতো না দেয়া, কোনো চলকের মান না জানানো প্রভৃতি। এসব ভুল সংশোধন করা খুবই সহজ, কারণ সিনট্যাক্স ভুল হলে অনুবাদক প্রোগ্রাম ভুলের বার্তা ছাপায়। যেমন- প্রোগ্রামে printf() কমান্ডের পরিবর্তে print() লেখা। সিনট্যাক্স ভুলকে কম্পাইল টাইম ভুলও বলা হয়।

লজিক্যাল বা যৌক্তিক ভুলঃ প্রোগ্রামে যুক্তির ভুল থাকলে তাকে বলে লজিক্যাল ভুল। সাধারণত সমস্যা ঠিকমতো না বুঝার জন্যই এ ভুল হয়। যেমন- a>b এর স্থলে a<b বা s=a+b এর স্থানে s=a-b লিখলে লজিক্যাল ভুল হয়। লজিক্যাল ভুলের ক্ষেত্রে একটি উত্তর পাওয়া যায় যদিও তা ভুল। এক্ষেত্রে অনুবাদক প্রোগ্রাম কোনো ভুলের বার্তা ছাপায় না বলে লজিক্যাল ভুল সংশোধন করা খুব কঠিন।

রান টাইম বা এক্সিকিউশন টাইম ভুলঃ রান টাইম ভুল প্রোগ্রাম এক্সিকিউশনের সময় ঘটে। যেমন- শূন্য দিয়ে ভাগ করা কিংবা ঋণাত্বক সংখ্যার বর্গমূল বা লগারিদম বের করা, ডাইনামিক মেমোরি অ্যালোকেশনের সময় অপর্যাপ্ত মেমোরি থাকা ইত্যাদি। অনুবাদক প্রোগ্রাম অনুবাদ করার সময় এই ধরণের ভুল নির্নয় করতে পারে না। এই ধরণের ভুল সম্বলিত প্রোগ্রাম রান করবে কিন্তু প্রোগ্রাম বন্ধ হয়ে যেতে পারে। রান টাইম ভুল নির্নয় এবং সংশোধন করা কঠিন। যৌক্তিক ভুল এক ধরণের রান-টাইম ভুল কারণ এই ধরণের ভুল কম্পাইলার নির্নয় করতে পারে না বা ডিবাগিং এর মাধ্যমে নির্নয় করা যায় না।

ডিবাগিংঃ আমরা প্রোগ্রাম টেস্টিং এর ক্ষেত্রে প্রোগ্রামে বিভিন্ন ধরনের ভুল সম্পর্কে জেনেছি। প্রোগ্রামে যেকোনো ভুল চিহ্নিত করতে পারলে সেই ভুলকে বলা হয় বাগ (Bug)। উক্ত ভুল বা Bug কে সমাধান করাকে বলা হয় ডিবাগ (Debug)। অর্থাৎ প্রোগ্রামের ভুল-ত্রুটি(Error) খুঁজে বের করে তা সমাধান করার প্রক্রিয়াকে বলা হয় ডিবাগিং।

এক্ষেত্রে ডিবাগিং এর মাধ্যমে Syntax Error সমাধান করা সহজ কিন্তু Logical Error এবং Run-time Error সমাধান করা তুলনামূলক জটিল।

১৯৪৫ সালে মার্ক-১ কম্পিউটারের ভিতরে একটি মথপোকা প্রবেশ করে বাসা বাধে, ফলে কম্পিউটারটি অকার্যকর হয়ে পড়ে। তখন থেকে কম্পিউটার বিজ্ঞানে Bug কথাটির অর্থ ত্রুটি(Error)।

প্রোগ্রাম রক্ষণাবেক্ষণঃ

সময়ের সাথে পরিবেশ-পরিস্থিতি পরিবর্তনের কারণে প্রোগ্রামের পরিবর্তন বা আধুনিকীকরণ করা প্রয়োজন হয়। নিমোক্ত আউটকাম অর্জনের জন্য প্রোগ্রাম রক্ষণাবেক্ষণ করার প্রয়োজন হয়-

১। ভুল সংশোধন

২। কর্মক্ষমতা বৃদ্ধি

৩। প্রোগ্রামের নতুন ফিচার যুক্ত করা

৪। অপ্রয়োজনীয় অংশ বাদ দেওয়া ইত্যাদি।

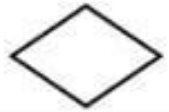
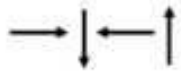
সূডোকোড কী? (Pseudo Code)

সূডো (Pseudo) একটি গ্রীক শব্দ যার অর্থ হচ্ছে ছদ্ম বা ফেইক(fake)। সূডোকোড শব্দটি বোঝায় এটি কোন কোড নয়।

সূডোকোড হল কোনও প্রোগ্রাম বর্ণনার একটি ইনফরমাল উপায় যাতে প্রোগ্রামিং ভাষার সিনট্যাক্স বা কোনও প্রযুক্তি ব্যবহৃত হয় না। এটি কোনও প্রোগ্রামের একটি রূপরেখা বা খসড়া তৈরির জন্য ব্যবহৃত হয়। সূডোকোড একটি প্রোগ্রাম প্রবাহের ধারণা দেয়, তবে বিস্তারিত কিছু প্রকাশ করে না। সিস্টেম ডিজাইনারগণ সূডোকোড লিখে যাতে প্রোগ্রামাররা কোনও সফ্টওয়্যার প্রকল্পের প্রয়োজনীয়তা বুঝতে পারে এবং সে অনুযায়ী কোড লিখতে পারেন।

একটি প্রোগ্রামের কার্যপ্রণালী বর্ণনা বা উপস্থাপনার জন্য ইনফরমাল উপায়ে ইংরেজি ভাষায় লেখা কতগুলো নির্দেশনার সমষ্টিকে একত্রে সূডোকোড বলে। নিচের উদাহরণটি লক্ষ্য কর-

ফ্লোচার্টে ব্যবহৃত প্রতিক সমূহ ও ব্যবহারঃ

চিহ্ন	চিহ্নের নাম	ব্যবহার
	Terminal / ডিম্বক	ফ্লোচার্টের শুরু এবং শেষ নির্দেশ করতে
	সামন্তরিক	ইনপুট নেওয়া ও আউটপুট প্রদর্শন করতে
	আয়তক্ষেত্র	প্রক্রিয়াকরণ নির্দেশ করতে
	হীরক	শর্ত সাপেক্ষে কোন প্রক্রিয়া সম্পন্ন করতে এই চিহ্নের মধ্যে শর্ত লেখা হয়
	বৃত্ত	একাদিক শাখা যখন একটি বিন্দুতে মিলিত হয় তখন এই প্রতীক ব্যবহৃত হয়
	তীর চিহ্ন	প্রবাহের দিক নির্দেশে ব্যবহৃত হয়

১। দুইটি সংখ্যা ইনপুট নিয়ে যোগফল নির্ণয় করার অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

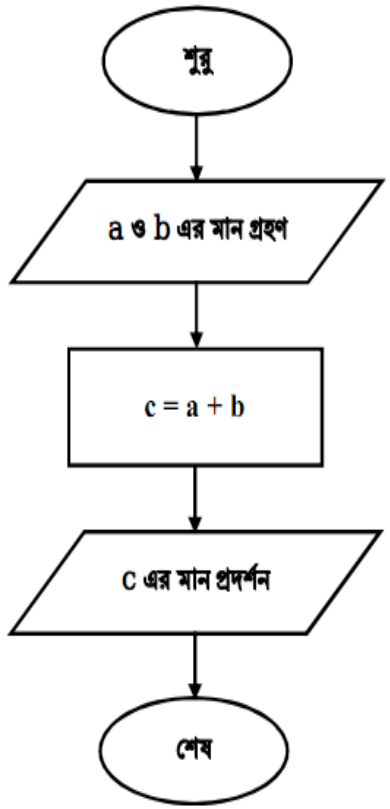
ধাপ-১: শুরু।

ধাপ-২: a ও b চলকের মান গ্রহণ।

ধাপ-৩: c =a+b নির্ণয়।

ধাপ-৪: c চলকের মান প্রদর্শন।

ধাপ-৫: শেষ।


শুরু
a ও b এর মান গ্রহণ
c = a + b
c এর মান প্রদর্শন
শেষ

২। দুইটি সংখ্যা ইনপুট নিয়ে বিয়োগফল নির্ণয় করার অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

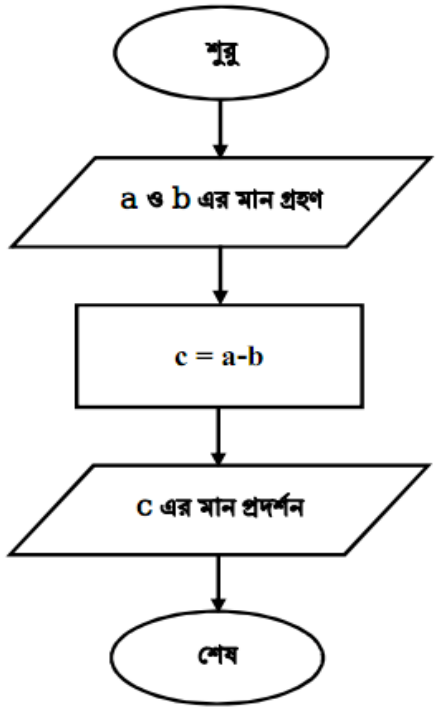
ধাপ-২: a ও b চলকের মান গ্রহণ।

ধাপ-৩: c = a-b নির্ণয়।

ধাপ-৪: c চলকের মান প্রদর্শন।

ধাপ-৫: শেষ।

ফ্লোচার্টঃ



৩। দুইটি সংখ্যা ইনপুট নিয়ে গুণফল নির্ণয় করার অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

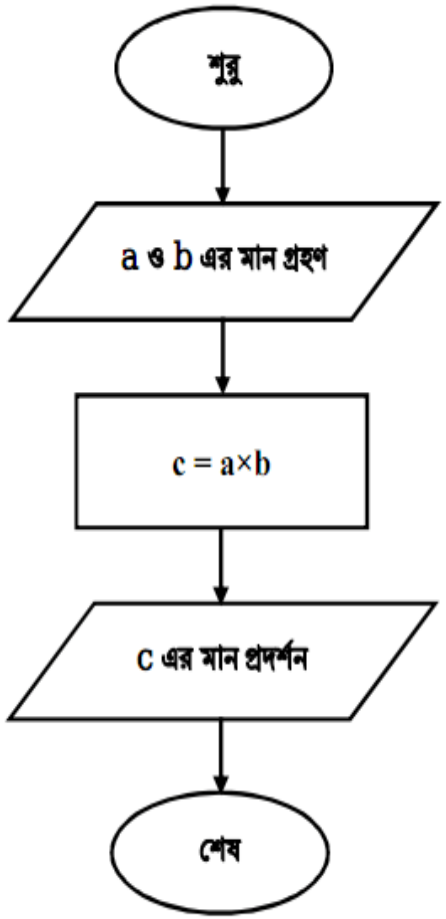
ধাপ-২: a ও b চলকের মান গ্রহণ।

ধাপ-৩: c=a×b নির্ণয়।

ধাপ-৪: c চলকের মান প্রদর্শন।

ধাপ-৫: শেষ।

ফ্লোচার্টঃ



৫। সেলসিয়াস স্কেলের তাপমাত্রাকে ফারেনহাইট স্কেলের তাপমাত্রায় রুপান্তরের অ্যালগোরিদম ও ফ্লোচার্ট।

তাপমাত্রা পরিমাপের বিভিন্ন স্কেলের মধ্যে সম্পর্ক- C/5 = F-32 / 9 = K-273 / 5

অ্যালগোরিদমঃ

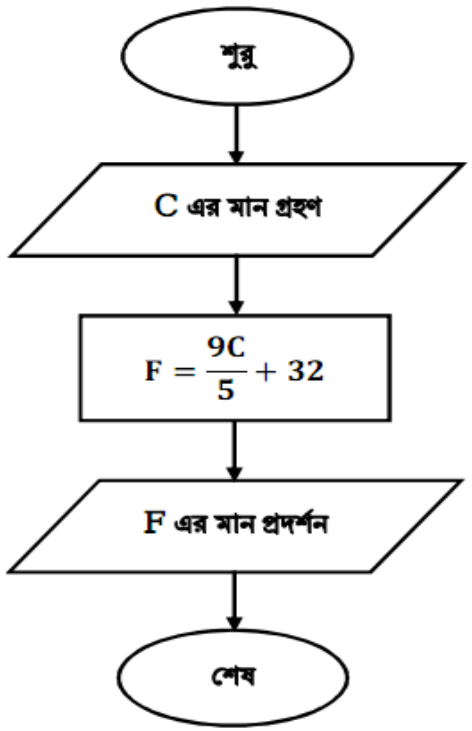
ধাপ-১: শুরু।

ধাপ-২: C চলকে সেলসিয়াস স্কেলের তাপমাত্রা গ্রহণ।

ধাপ-৩: **F = (9C/5)+32** নির্ণয়।

ধাপ-৪: F চলকের মান প্রদর্শন।

ধাপ-৫: শেষ।


শুরু
C এর মান গ্রহণ
$F = \frac{9C}{5} + 32$
F এর মান প্রদর্শন
শেষ

৬। ফারেনহাইট স্কেলের তাপমাত্রাকে সেলসিয়াস তাপমাত্রায় রুপান্তরের অ্যালগোরিদম ও ফ্লোচার্ট।

তাপমাত্রা পরিমাপের বিভিন্ন স্কেলের মধ্যে সম্পর্ক- C/5 = F-32 / 9 = K-273 / 5

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

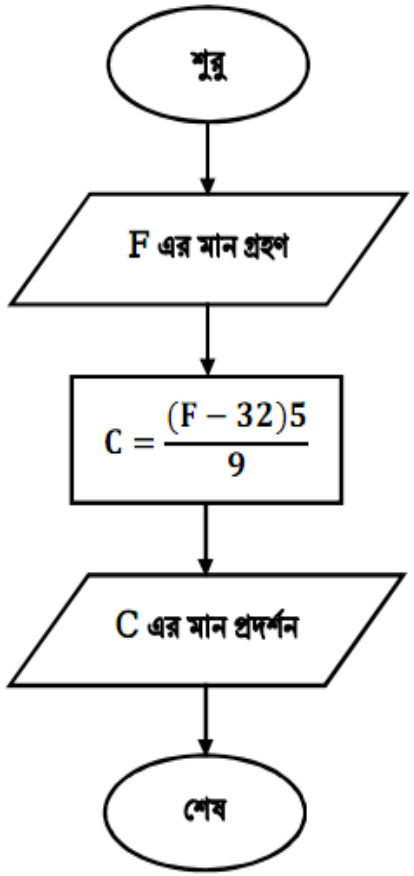
ধাপ-২: F চলকে ফারেনহাইট স্কেলের তাপমাত্রা গ্রহণ।

ধাপ-৩: **C = ((F-32)5)/9** নির্ণয়।

ধাপ-৪: C চলকের মান প্রদর্শন।

ধাপ-৫: শেষ।

ফ্লোচার্টঃ



৭। ত্রিভুজের ভুমি ও উচ্চতা দেওয়া থাকলে ক্ষেত্রফল নির্ণয়ের অ্যালগোরিদম ও ফ্লোচার্ট।

ত্রিভুজের ভুমি ও উচ্চতা দেওয়া থাকলে ক্ষেত্রফল নির্ণয়ের সূত্র, ক্ষেত্রফল =½×ভূমি×উচ্চতা।

অ্যালগোরিদমঃ

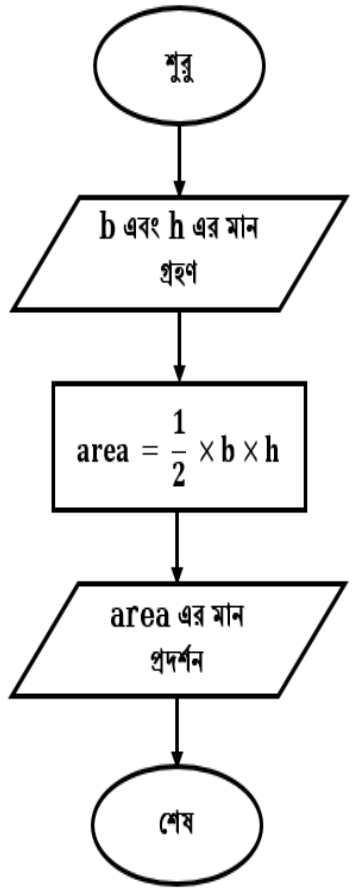
ধাপ-১: শুরু।

ধাপ-২: b এবং h চলকে যথাক্রমে ত্রিভুজের ভূমি ও উচ্চতার মান গ্রহণ।

ধাপ-৩: **area = ½×b×h** নির্ণয়।

ধাপ-৪: area চলকের মান প্রদর্শন।

ধাপ-৫: শেষ।


শুরু
b এবং h এর মান
গ্রহণ
area = 1/2 × b × h
area এর মান
প্রদর্শন
শেষ

৮। ত্রিভুজের তিনটি বাহুর দৈর্ঘ্য যথাক্রমে a, b এবং c দেওয়া থাকলে ক্ষেত্রফল নির্ণয়ের অ্যালগোরিদম ও ফ্লোচার্ট।

ত্রিভুজের তিনটি বাহুর দৈর্ঘ্য যথাক্রমে a, b এবং c দেওয়া থাকলে ক্ষেত্রফল নির্ণয়ের সূত্র

---

area= $\sqrt{s(s-a)(s-b)(s-c)}$ [এখানে s=অর্ধপরিসীমা]

অর্ধপরিসীমা s=(a+b+c)/2

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

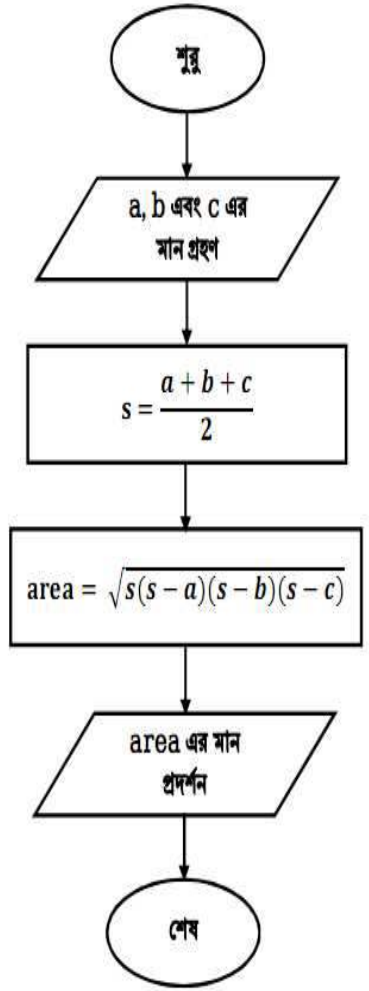
ধাপ-২: a, b এবং c চলকে ত্রিভুজের তিন বাহুর মান গ্রহণ।

ধাপ-৩: **s = (a+b+c)/2** নির্ণয়।

ধাপ-৪: **area = √s(s-a)(s-b)(s-c)** নির্ণয়।

ধাপ-৫: area চলকের মান প্রদর্শন।

ধাপ-৬: শেষ।


শুরু
a, b এবং c এর মান গ্রহণ
$s = \frac{a+b+c}{2}$
$\text{area} = \sqrt{s(s-a)(s-b)(s-c)}$
area এর মান প্রদর্শন
শেষ

১০। বৃত্তের ক্ষেত্রফল নির্ণয়ের অ্যালগোরিদম ও ফ্লোচার্ট।

বৃত্তের ব্যাসার্ধের মান দেওয়া থাকলে ক্ষেত্রফল নির্ণয়ের সূত্র, ক্ষেত্রফল=πr^2

অ্যালগোরিদমঃ

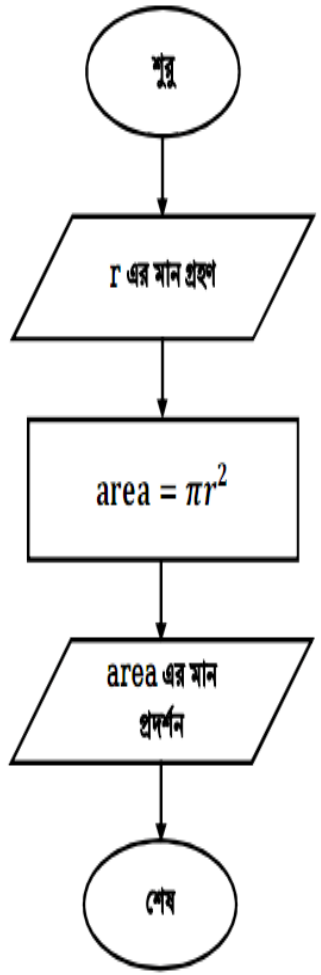
ধাপ-১: শুরু।

ধাপ-২: r চলকে বৃত্তের ব্যাসার্ধের মান গ্রহণ।

ধাপ-৩: **area= πr^2** নির্ণয়।

ধাপ-৪: area চলকের মান প্রদর্শন।

ধাপ-৫: শেষ।


শুরু
r এর মান গ্রহণ
area = πr^2
area এর মান
প্রদর্শন
শেষ

৭। তিনটি সংখ্যার মধ্যে সবচেয়ে ছোট সংখ্যা নির্ণয়ের অ্যালগোরিদম ও ফ্লোচার্ট ।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

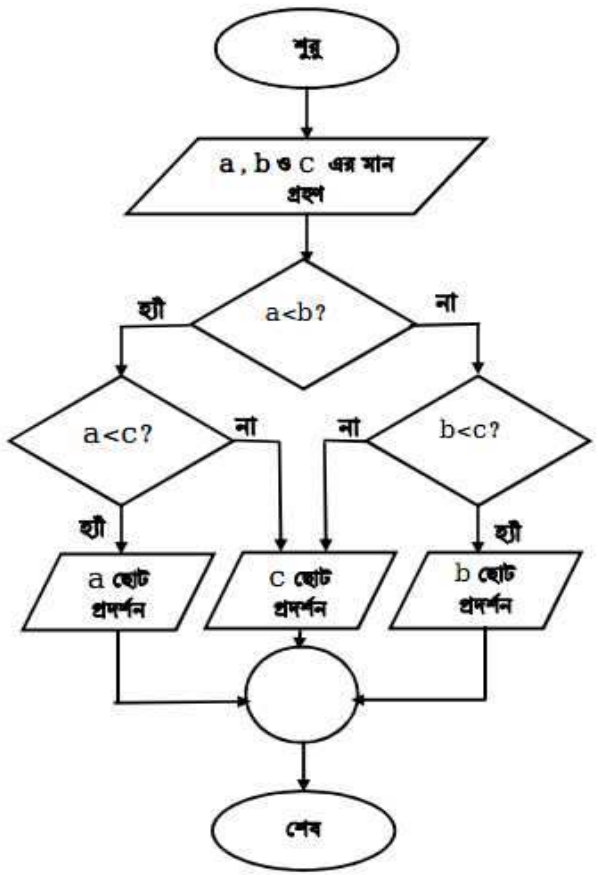
ধাপ-২: a, b ও c চলকে তিনটি সংখ্যা গ্রহণ।

ধাপ-৩: যদি a<b হয়, তাহলে ৪ নং ধাপে যাই, অন্যথায় ৫নং ধাপে যাই।

ধাপ-৪: যদি a<c হয়, তাহলে a ছোট প্রদর্শন এবং ৬নং ধাপে যাই, অন্যথায় c ছোট প্রদর্শন এবং ৬নং ধাপে যাই।

ধাপ-৫: যদি b<c হয়, তাহলে b ছোট প্রদর্শন, অন্যথায় c ছোট প্রদর্শন।

ফ্লোচার্টঃ


শুরু
a , b ও c এর মান গ্রহণ
a<b?
হ্যাঁ
না
a<c?
b<c?
না
না
হ্যাঁ
হ্যাঁ
a ছোট প্রদর্শন
c ছোট প্রদর্শন
b ছোট প্রদর্শন
শেষ

৮। তিনটি সংখ্যার মধ্যে সবচেয়ে বড় সংখ্যা নির্ণয়ের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২: a, b ও c চলকে তিনটি সংখ্যা গ্রহণ।

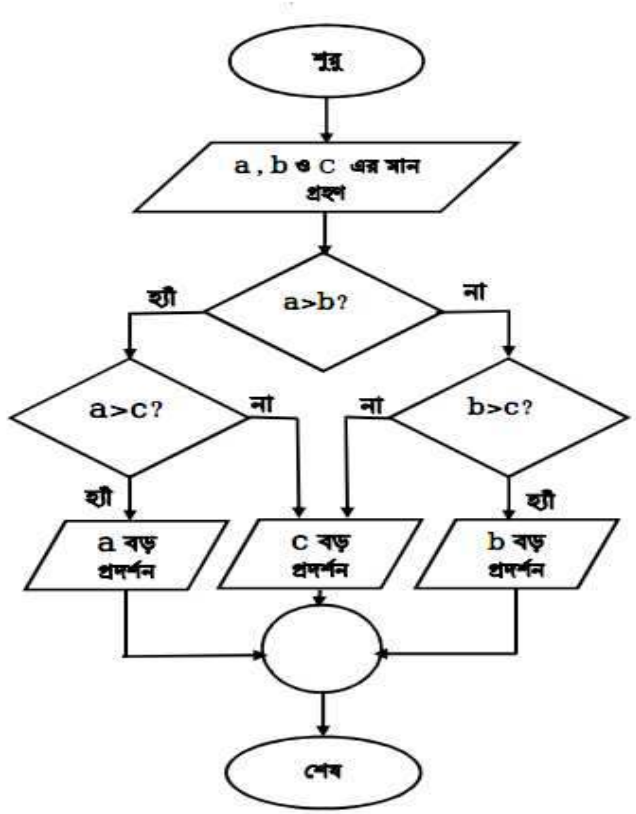
ধাপ-৩: যদি a>b হয়, তাহলে ৪ নং ধাপে যাই, অন্যথায় ৫নং ধাপে যাই।

ধাপ-৪: যদি a>c হয়, তাহলে a বড় প্রদর্শন এবং ৬নং ধাপে যাই, অন্যথায় c বড় প্রদর্শন এবং ৬নং ধাপে যাই।

ধাপ-৫: যদি b>c হয়, তাহলে b বড় প্রদর্শন, অন্যথায় c বড় প্রদর্শন।

ধাপ-৬: শেষ।

ফ্লোচার্টঃ


শুরু
a , b ও c এর মান গ্রহণ
a>b?
হ্যাঁ
না
a>c?
না
হ্যাঁ
b>c?
না
হ্যাঁ
a বড় প্রদর্শন
c বড় প্রদর্শন
b বড় প্রদর্শন
শেষ

১। ১ থেকে ১০ পর্যন্ত সংখ্যা দেখানোর অ্যালগোরিদম ও ফ্লোচার্ট। অথবা ১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯ ১০ ক্রম প্রদর্শনের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু ।

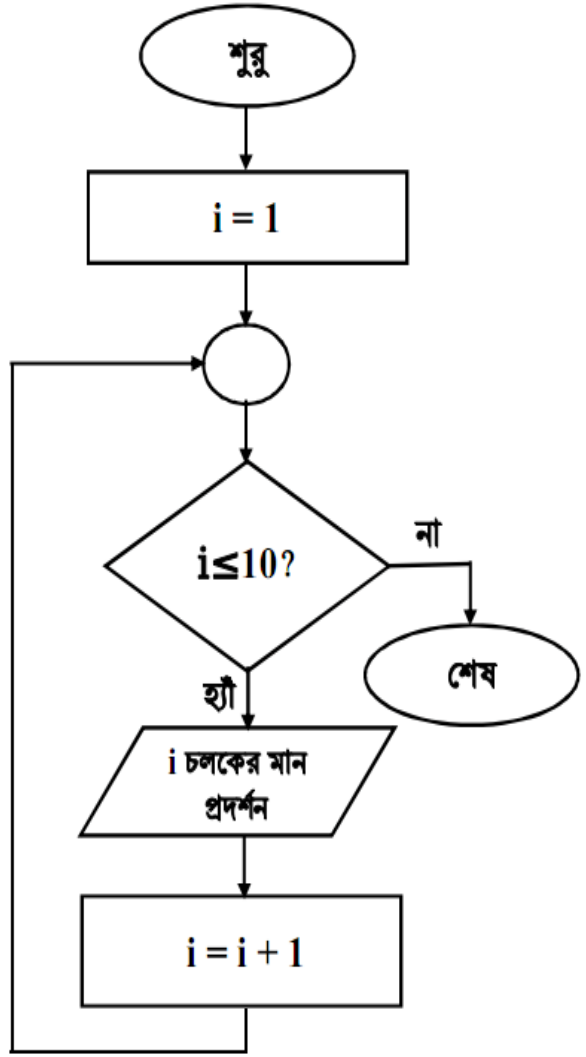
ধাপ-২: i চলকের মান ১ দ্বারা সূচনা করি।

ধাপ-৩: যদি i ≤ 10 হয়, তাহলে ৪নং ধাপে যাই, অন্যথায় ৬নং ধাপে যাই।

ধাপ-৪: i চলকের মান প্রদর্শন।

ধাপ-৫: i চলকের মান ১ বৃদ্ধি করি এবং পুনরায় ৩নং ধাপে যাই।

ধাপ-৬: শেষ।


শুরু
i = 1
i≤10?
না
শেষ
হ্যাঁ
i চলকের মান
প্রদর্শন
i = i + 1

২। ১ থেকে n পর্যন্ত সংখ্যা দেখানোর অ্যালগোরিদম ও ফ্লোচার্ট। অথবা ১ ২ ৩ ৪ ৫ – – – – – n ক্রম প্রদর্শনের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২: n চলকের মান গ্রহণ।

ধাপ-৩: i চলকের মান ১ দ্বারা সূচনা করি।

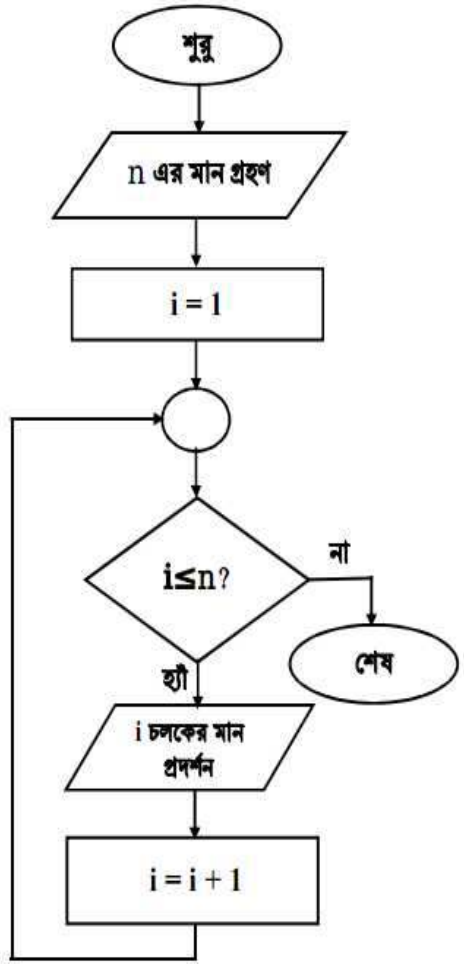
ধাপ-৪: যদি i ≤ n হয়, তাহলে ৫নং ধাপে যাই, অন্যথায় ৭নং ধাপে যাই।

ধাপ-৫: i চলকের মান প্রদর্শন।

ধাপ-৬: i চলকের মান ১ বৃদ্ধি করি এবং পুনরায় ৪নং ধাপে যাই।

ধাপ-৭: শেষ।

ফ্লোচার্টঃ



৩। ১ থেকে ১০ এর মধ্যে অবস্থিত বিজোড় সংখ্যাগুলো দেখানোর অ্যালগোরিদম ও ফ্লোচার্ট। অথবা

১ ৩ ৫ ৭ ৯ ক্রম প্রদর্শনের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২: i চলকের মান ১ দ্বারা সূচনা করি।

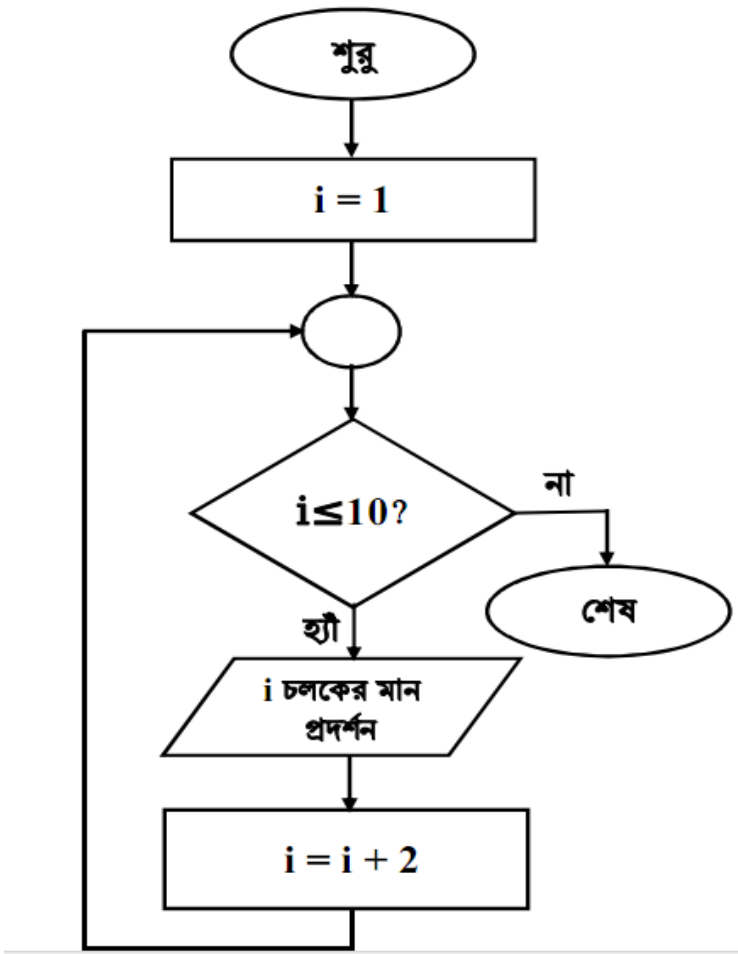
ধাপ-৩: যদি i ≤ 10 হয়, তাহলে ৪নং ধাপে যাই, অন্যথায় ৬নং ধাপে যাই।

ধাপ-৪: i চলকের মান প্রদর্শন।

ধাপ-৫: i চলকের মান ২ বৃদ্ধি করি এবং পুনরায় ৩নং ধাপে যাই।

ধাপ-৬: শেষ।

ফ্লোচার্টঃ



৪। ১ থেকে n এর মধ্যে অবস্থিত বিজোড় সংখ্যাগুলো দেখানোর অ্যালগোরিদম ও ফ্লোচার্ট। অথবা

১ ৩ ৫ ৭ - - -n ক্রম প্রদর্শনের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২:n চলকের মান গ্রহণ।

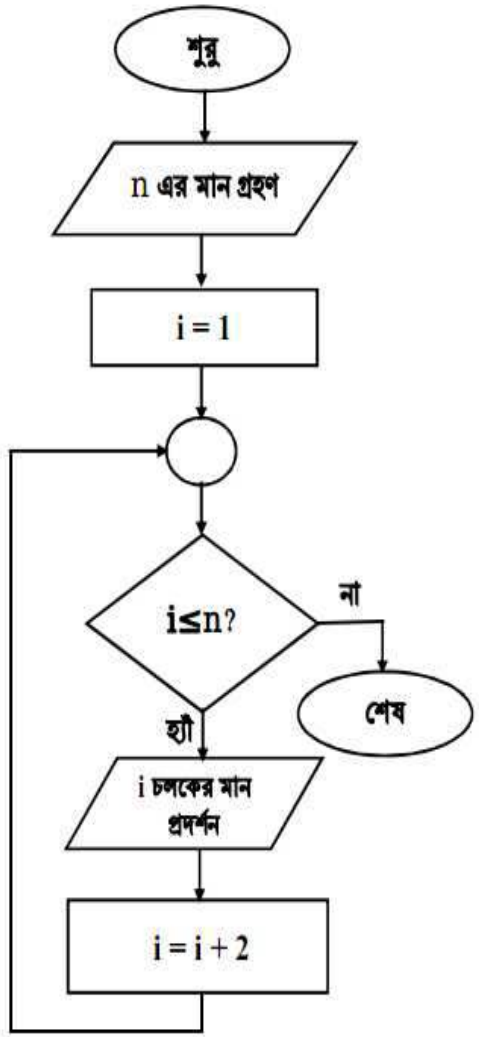
ধাপ-৩: i চলকের মান ১ দ্বারা সূচনা করি।

ধাপ-৪: যদি i ≤ n হয়, তাহলে ৫নং ধাপে যাই, অন্যথায় ৭নং ধাপে যাই।

ধাপ-৫: i চলকের মান প্রদর্শন।

ধাপ-৬: i চলকের মান ২ বৃদ্ধি করি এবং পুনরায় ৪নং ধাপে যাই।

ধাপ-৭: শেষ।


শুরু
n এর মান গ্রহণ
i = 1
i≤n?
না
শেষ
হ্যাঁ
i চলকের মান
প্রদর্শন
i = i + 2

৫। ১ থেকে ১০ এর মধ্যে অবস্থিত জোড় সংখ্যাগুলো দেখানোর অ্যালগোরিদম ও ফ্লোচার্ট। অথবা

২ ৪ ৬ ৮ ১০ ক্রম প্রদর্শনের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২: i চলকের মান ২ দ্বারা সূচনা করি।

ধাপ-৩: যদি i ≤ 10 হয়, তাহলে ৪নং ধাপে যাই, অন্যথায় ৬নং ধাপে যাই।

ধাপ-৪: i চলকের মান প্রদর্শন।

ধাপ-৫: i চলকের মান ২ বৃদ্ধি করি এবং পুনরায় ৩নং ধাপে যাই।

ধাপ-৬: শেষ।

৫। ১ থেকে ১০ এর মধ্যে অবস্থিত জোড় সংখ্যাগুলো দেখানোর অ্যালগোরিদম ও ফ্লোচার্ট। অথবা

২ ৪ ৬ ৮ ১০ ক্রম প্রদর্শনের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

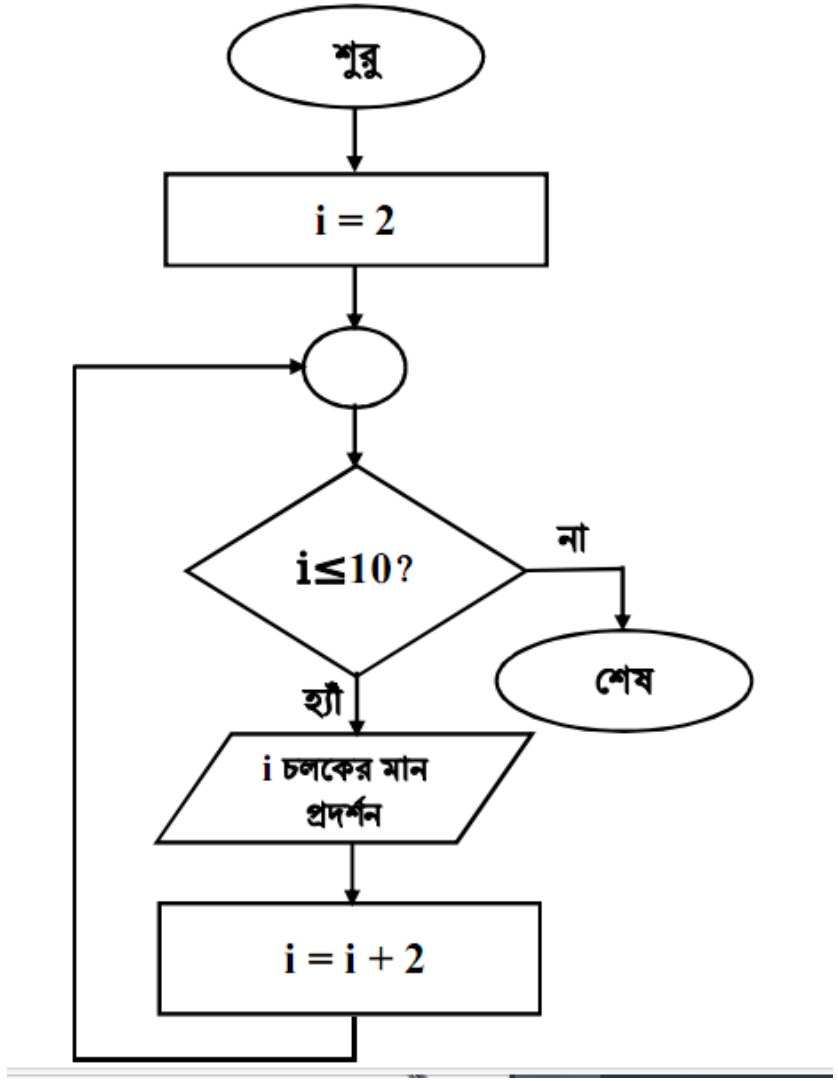
ধাপ-২: i চলকের মান ২ দ্বারা সূচনা করি।

ধাপ-৩: যদি i ≤ 10 হয়, তাহলে ৪নং ধাপে যাই, অন্যথায় ৬নং ধাপে যাই।

ধাপ-৪: i চলকের মান প্রদর্শন।

ধাপ-৫: i চলকের মান ২ বৃদ্ধি করি এবং পুনরায় ৩নং ধাপে যাই।

ধাপ-৬: শেষ।


শুরু
i = 2
i≤10?
না
শেষ
হ্যাঁ
i চলকের মান
প্রদর্শন
i = i + 2

৬। ১ থেকে n এর মধ্যে অবস্থিত জোড় সংখ্যাগুলো দেখানোর অ্যালগোরিদম ও ফ্লোচার্ট। অথবা

২ ৪ ৬ – – – n ক্রম প্রদর্শনের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২: n চলকের মান গ্রহণ।

ধাপ-৩: i চলকের মান ২ দ্বারা সূচনা করি।

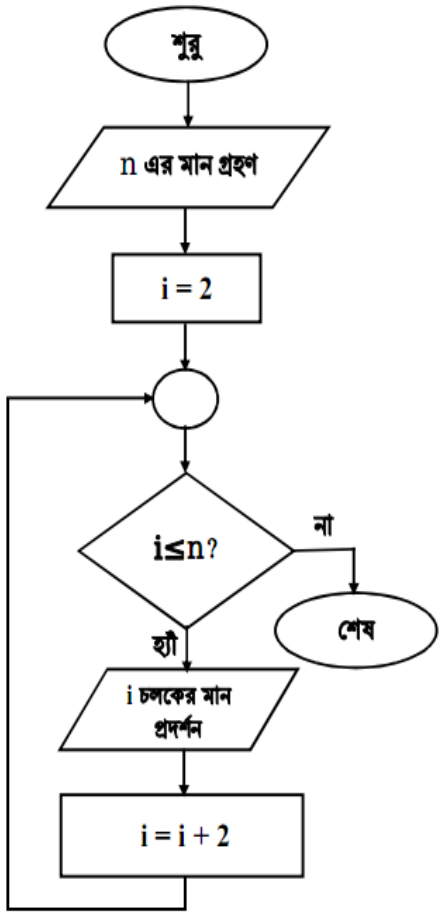
ধাপ-৪: যদি i ≤ n হয়, তাহলে ৫নং ধাপে যাই, অন্যথায় ৭নং ধাপে যাই।

ধাপ-৫: i চলকের মান প্রদর্শন।

ধাপ-৬: i চলকের মান ২ বৃদ্ধি করি এবং পুনরায় ৪নং ধাপে যাই।

ধাপ-৭: শেষ।

ফ্লোচার্টঃ



৭। ১ থেকে ১০০ পর্যন্ত সংখ্যা গুলোর যোগফল নির্নয়ের অ্যালগোরিদম ও ফ্লোচার্ট। অথবা

১+২+৩+ – – – – -+১০০ ধারার যোগফল নির্নয়ের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২: i চলকের মান 1 দ্বারা এবং sum চলকের মান ০ দ্বারা সূচনা করি।

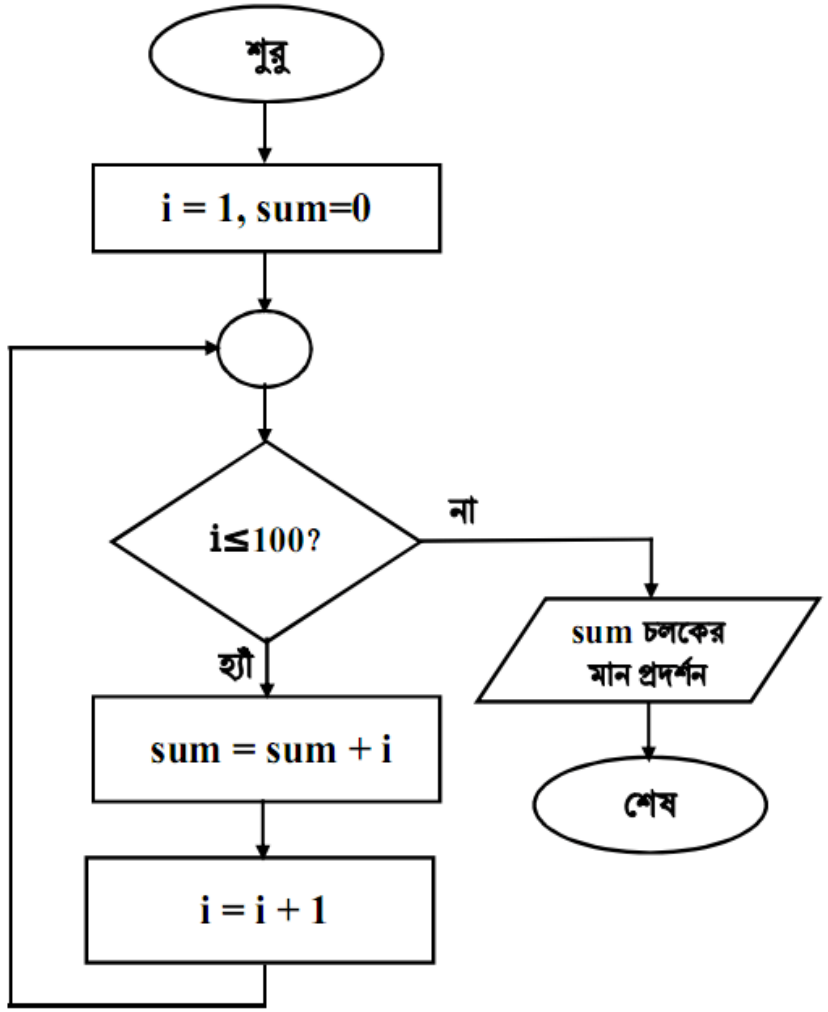
ধাপ-৩: যদি i ≤ 100 হয়, তাহলে ৪নং ধাপে যাই, অন্যথায় ৬নং ধাপে যাই।

ধাপ-৪: sum = sum + i নির্ণয়।

ধাপ-৫: i চলকের মান 1 বৃদ্ধি করি এবং পুনরায় ৩নং ধাপে যাই।

ধাপ-৬: sum চলকের মান প্রদর্শন।

ধাপ-৭: শেষ।


শুরু
i = 1, sum=0
i≤100?
না
হ্যাঁ
sum চলকের
মান প্রদর্শন
sum = sum + i
শেষ
i = i + 1

৮। ১ থেকে n পর্যন্ত সংখ্যা গুলোর যোগফল নির্নয়ের অ্যালগোরিদম ও ফ্লোচার্ট। অথবা

১+২+৩+ – – – – -+n ধারার যোগফল নির্নয়ের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২: n চলকের মান গ্রহণ করি।

ধাপ-৩: i চলকের মান 1 দ্বারা এবং sum চলকের মান ০ দ্বারা সূচনা করি।

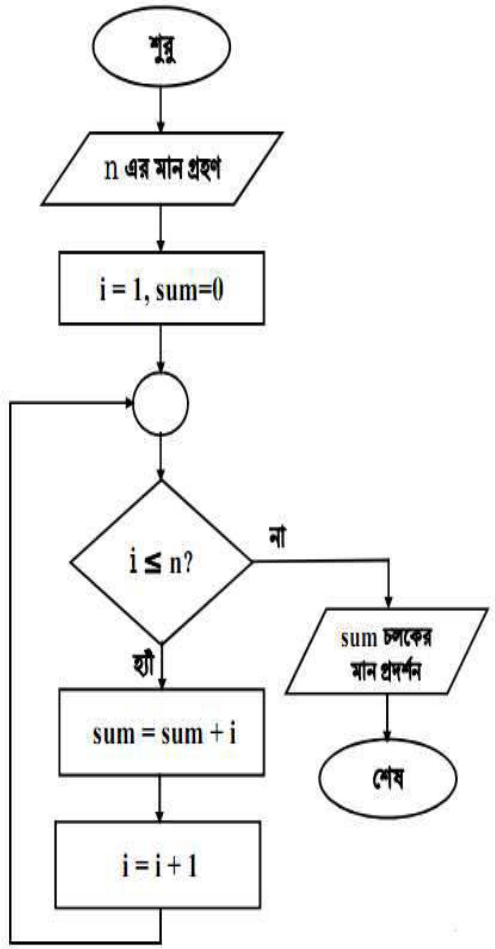
ধাপ-৪: যদি i ≤ n হয়, তাহলে ৫নং ধাপে যাই, অন্যথায় ৭নং ধাপে যাই।

ধাপ-৫: sum = sum + i নির্ণয়।

ধাপ-৬: i চলকের মান 1 বৃদ্ধি করি এবং পুনরায় ৪নং ধাপে যাই।

ধাপ-৭: sum চলকের মান প্রদর্শন।

ধাপ-৮: শেষ।


শুরু
n এর মান গ্রহণ
i = 1, sum=0
i ≤ n?
না
হ্যাঁ
sum চলকের মান প্রদর্শন
sum = sum + i
শেষ
i = i + 1

৯। ১ থেকে ১০০ এর মধ্যে অবস্থিত বিজোড় সংখ্যা গুলোর যোগফল নির্নয়ের অ্যালগোরিদম ও ফ্লোচার্ট। অথবা

১+৩+৫+ – – – – -+১০০ ধারার যোগফল নির্নয়ের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২: i চলকের মান 1 দ্বারা এবং sum চলকের মান ০ দ্বারা সূচনা করি।

ধাপ-৩: যদি i ≤ 100 হয়, তাহলে ৪নং ধাপে যাই, অন্যথায় ৬নং ধাপে যাই।

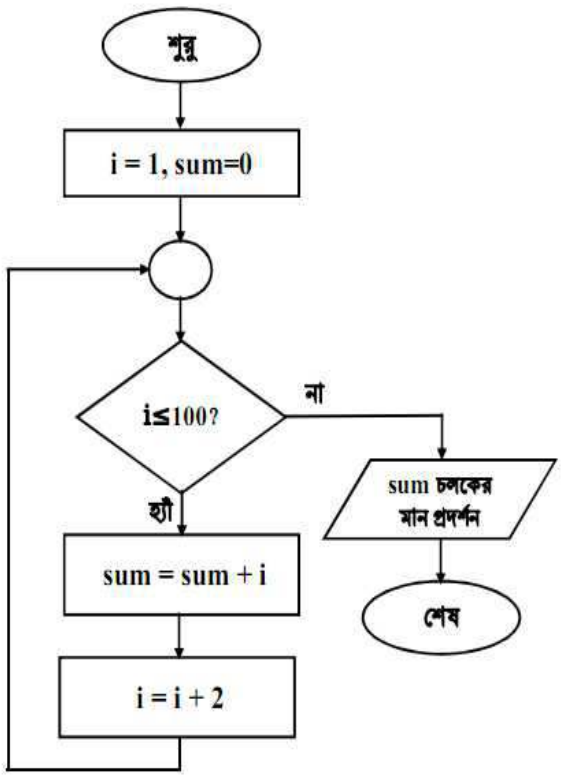
ধাপ-৪: sum = sum + i নির্ণয়।

ধাপ-৫: i চলকের মান ২ বৃদ্ধি করি এবং পুনরায় ৩নং ধাপে যাই।

ধাপ-৬: sum চলকের মান প্রদর্শন।

ধাপ-৭: শেষ।

ফ্লোচার্টঃ



১০। ১ থেকে ১০০ এর মধ্যে অবস্থিত জোড় সংখ্যা গুলোর যোগফল নির্নয়ের অ্যালগোরিদম ও ফ্লোচার্ট। অথবা

২+৪+৬+ – – – – -+১০০ ধারার যোগফল নির্নয়ের অ্যালগোরিদম ও ফ্লোচার্ট।

অ্যালগোরিদমঃ

ধাপ-১: শুরু।

ধাপ-২: i চলকের মান ২ দ্বারা এবং sum চলকের মান ০ দ্বারা সূচনা করি।

ধাপ-৩: যদি i ≤ 100 হয়, তাহলে ৪নং ধাপে যাই, অন্যথায় ৬নং ধাপে যাই।

ধাপ-৪: sum = sum + i নির্ণয়।

ধাপ-৫: i চলকের মান ২ বৃদ্ধি করি এবং পুনরায় ৩নং ধাপে যাই।

ধাপ-৬: sum চলকের মান প্রদর্শন।

ধাপ-৭: শেষ।

Chapter-2
Data types, constants & variables

After completing this chapter we will learn about-

- What is data types
- Data type declare how
- Constant, variable & string

ডেটা টাইপ (Data Type):

সি প্রোগ্রামে বিভিন্ন ধরনের ডেটা নিয়ে কাজ করা হয়। প্রোগ্রামে সাধারণত ডেটা ইনপুট করা হয় এবং প্রোগ্রাম ডেটা প্রসেস করে আউটপুট প্রদান করে থাকে। সি-প্রোগ্রামিং-এ ভেরিয়েবল ঘোষণার জন্য ডেটার মান অনুযায়ী ডেটা টাইপ বলে দিতে হয়। বিভিন্ন টাইপের ডেটা মেমোরিতে ভিন্ন ভিন্ন বাইটের জায়গা দখল করে। ডেটার ধরন ও মেমোরি পরিসর সংরক্ষণের ভিত্তিতে সি ভাষায় চারটি মৌলিক ডেটা টাইপ আছে। যথা-

1. ক্যারেক্টার (Character)
2. ইন্টেজার (Integer)
3. ফ্লোটিং পয়েন্ট (Floating Point)
4. ডাবল (Double)

ডেটা টাইপ ঘোষণার নিয়ম:

- ভেরিয়েবল ঘোষণার আগে ডেটা টাইপ নির্ধারণ করার জন্য ডেটা টাইপের কী-ওয়ার্ড লিখতে হবে। যেমন- int, float, char, double ইত্যাদি।
- তারপর স্পেস দিয়ে ভেরিয়েবল-এর নাম লিখতে হবে। যেমন- int sum
- সবশেষে সেমিকোলন দিতে হবে। যেমন- int sum;
- একই টাইপের একাধিক ভেরিয়েবল ঘোষণায় ডেটা টাইপ কী-ওয়ার্ডের পর একই লাইনে কমা দিয়ে ভেরিয়েবলসমূহকে আলাদা করে লিখে সবশেষে সেমিকোলন লিখতে হবে। যেমন- int n1, n2, sum ইত্যাদি।

ডেটা টাইপ	ব্যবহৃত কী ওয়ার্ড	উদাহরণ
Character	char	char a; char a,b,c;
Integer	int	int number; int number1,number2
Float	float	float x; float a,b,c;
Double	double	double x; double x,y,z;

বিভিন্ন ধরনের ডেটা টাইপ, সাইজ এবং ভেরিয়েবলের রেঞ্জ:

ডেটা টাইপ	সাইজ	ভেরিয়েবলের রেঞ্জ
char	1 byte	-128 to 127 বা, -2^{7} to 2^{7-1}
signed char	1 byte	-128 to 127 বা, -2^{7} to 2^{7-1}
unsigned char	1 byte	0 to 255 বা, 0 to (2^{8-1})
int	2/4 byte	-32768 to 32767 বা, -2^{15} to 2^{15-1}
signed int	2 byte	-32768 to 32767 বা, -2^{15} to 2^{15-1}
unsigned int	2 byte	0 to 65535 বা, 0 to (2^{16-1})
short int	2 byte	-32768 to 32767
enum	2 byte	-32768 to 32767
long int	4 byte	-2147483648 to 2147483647 বা, -2^{31} to 2^{31-1}
signed long	4 byte	-2147483648 to 2147483647
unsigned long	4 byte	0 to 4294967295 বা, 0 to (2^{32-1})
float	4 byte	3.4xE-38 to 3.4xE+38
double	8 byte	1.7xE-308 to 1.7xE+308
long double	10 byte	3.4xE-4932 to 1.1xE+4932
bool	1 bit	True or False

ডেটা টাইপ ঘোষণার নিয়ম:

- ভেরিয়েবল ঘোষণার আগে ডেটা টাইপ নির্ধারণ করার জন্য ডেটা টাইপের কী-ওয়ার্ড লিখতে হবে। যেমন- int, float, char, double ইত্যাদি।
- তারপর স্পেস দিয়ে ভেরিয়েবল-এর নাম লিখতে হবে। যেমন- int sum
- সবশেষে সেমিকোলন দিতে হবে। যেমন- int sum;
- একই টাইপের একাধিক ভেরিয়েবল ঘোষণায় ডেটা টাইপ কী-ওয়ার্ডের পর একই লাইনে কমা দিয়ে ভেরিয়েবলসমূহকে আলাদা করে লিখে সবশেষে সেমিকোলন লিখতে হবে। যেমন- int n1, n2, sum ইত্যাদি।

ডেটা টাইপ	ব্যবহৃত কী ওয়ার্ড	উদাহরণ
Character	char	char a; char a,b,c;
Integer	int	int number; int number1,number2
Float	float	float x; float a,b,c;
Double	double	double x; double x,y,z;

Constants (কনস্ট্যান্ট): প্রোগ্রাম নির্বাহের সময় "সি" প্রোগ্রামিং ভাষায় এমন কিছু মান আছে যা কখনো পরিবর্তন হয় না। যেমন π এর মান হলো বা ৩.১৪১৬ যা কখনো পরিবর্তন হয় না। প্রোগ্রাম নির্বাহের সময় যে রাশির মান অপরিবর্তীত থাকে তাকে কনস্ট্যান্ট বা ধ্রুবক বলে।

'সি' প্রোগ্রামিং ভাষায় দুইভাবে কনস্ট্যান্ট ঘোষণা করা যায়। যথা-

- ১। const কীওয়ার্ড ব্যবহার করে
- ২। #define প্রিপ্রসেসর ব্যবহার করে

const কীওয়ার্ড ব্যবহার করে ধ্রুবক ঘোষণার ফরম্যাট হলো:

- const ConstType ConstName = ConstValue;
- যেমনঃ const float PI=3.1416;

#define প্রিপ্রসেসর ব্যবহার করে ধ্রুবক ঘোষণার ফরম্যাট হলো:

- #define ConstName ConstValue
- #define PI 3.1416

স্ট্রিং(Strings): স্ট্রিং হলো কতগুলো ক্যারেক্টারের সমষ্টি যার শেষ উপাদান হলো null ক্যারেক্টার(\0)। এই null ক্যারেক্টার স্ট্রিং এর শেষ নির্দেশ করে। স্ট্রিং সবসময় ডাবল কোটেশনের (" ") সাহায্যে আবদ্ধ থাকে।

স্ট্রিং(Strings): স্ট্রিং হলো কতগুলো ক্যারেক্টারের সমষ্টি যার শেষ উপাদান হলো null ক্যারেক্টার(\0)। এই null ক্যারেক্টার স্ট্রিং এর শেষ নির্দেশ করে। স্ট্রিং সবসময় ডাবল কোটেশনের (" ") সাহায্যে আবদ্ধ থাকে।

স্ট্রিং ডিক্লারেশন করার পদ্ধতিঃ

- char string[20] = {'s','t','u','d','y', '\0'};
- char string[20] = "demo";
- char string [] = "demo";

Chapter-3
Operatorsts & expressions

After completing this chapter we will learn about-

- What is operator
- Different types of operator

অপারেটর (Operator):

প্রোগ্রামিং ভাষায় গাণিতিক ও যৌক্তিক কাজ নিয়ন্ত্রণ করার জন্য কতগুলো বিশেষ চিহ্ন ব্যবহৃত হয়, এগুলোকে অপারেটর বলা হয়। যেমন, +, -, *, /, <, > ইত্যাদি। এই অপারেটরসমূহ না থাকলে সি ভাষায় কখনো গাণিতিক বা যৌক্তিক কাজ করা সম্ভব হতো না। নিচে বিভিন্ন ধরনের অপারেটর সম্পর্কে আলোচনা করা হলো:

অ্যারিথমেটিক অপারেটর

সি প্রোগ্রামে বিভিন্ন গাণিতিক কাজ (যেমন: যোগ, বিয়োগ, গুণ, ভাগ ইত্যাদি) সম্পন্ন করার জন্য যেসব সিম্বল বা অপারেটর ব্যবহৃত হয়, তাদেরকে অ্যারিথমেটিক বা গাণিতিক অপারেটর বলা হয়। নিচের ছকে বহুল ব্যবহৃত অ্যারিথমেটিক অপারেটরসমূহের তালিকা ও ব্যবহার উল্লেখ করা হলো।

অপারেটর	ব্যবহার	উদাহরণ	ফলাফল
/ (ডিভিশন)	ভাগ করার জন্য।	81/9	9
* (মাল্টিপ্লিকেশন)	গুণ করার জন্য।	7*6	42
% (মডিউলো)	ভাগশেষ নির্ণয় ।	20%6	2
+ (বাইনারি প্লাস)	যোগ করা	8+5+2.75	15.75
- (বাইনারি মাইনাস)	বিয়োগ করা	9-5	4

টেবিল : অ্যারিথমেটিক অপারেটর

রিলেশনাল অপারেটর

সি প্রোগ্রামে দু'টো অপারেন্ডের মধ্যে সম্পর্ক (যেমন: ছোট, ছোট বা সমান, বড়, বড় বা সমান ইত্যাদি) বোঝানোর জন্য যে অপারেটর ব্যবহৃত হয়, তাকে রিলেশনাল অপারেটর বলা হয়। বস্তুত এই অপারেটর সংশ্লিষ্ট দুই অপারেন্ডের মধ্যে তুলনা করে ফলাফল সত্য/মিথ্যা হিসেবে জানিয়ে দেয়। নিম্নের ছকে বহুল ব্যবহৃত রিলেশনাল অপারেটরসমূহের তালিকা ও ব্যবহার উল্লেখ করা হলো:

অপারেটর	অর্থ
<	Less than (ছোট)
<=	Less than or equal (ছোট বা সমান)
>	Greater than (বড়)
>=	Greater than or equal (বড় বা সমান)
==	Equal to (সমান)
!=	Not equal to (অসমান)

টেবিল : রিলেশনাল অপারেটর

লজিক্যাল অপারেটর

সি প্রোগ্রামে লজিক্যাল অপরেশন (যেমন-লজিক্যাল অর, অজিক্যাল অ্যান্ড, অজিক্যাল নট) সম্পন্ন করার জন্য যে অপারেটর ব্যবহৃত হয়, তাকে লজিক্যাল অপারেটর বলা হয়। লজিক্যাল অর ও লজিক্যাল অ্যান্ড অপারেটর হলো বাইনারি অপারেটর এবং এদের উভয় অপারেন্ড int টাইপ হয়। লজিক্যাল অর (| |) ও লজিক্যাল অ্যান্ড (&&) – এর জন্য দু'টি করে অপারেন্ড থাকে, কিন্তু লজিক্যাল নট (!) এর জন্য একটি অপারেন্ড থাকে। এজন্য লজিক্যাল নট (!) একটি ইউনারি অপারেটর।

অপারেটর	উদাহরণ	ব্যবহার
|| (লজিক্যাল অর)	Op1=Op2)|| Op3;	দু'টি অপারেন্ডের মধ্যে লজিক্যাল OR অপারেশনের জন্য।
&& (লজিক্যাল অ্যান্ড)	Op1=Op2& &Op3;	দু'টি অপারেন্ডের মধ্যে লজিক্যাল AND অপারেশনের জন্য।
! * (লজিক্যাল নট)	Op1=!Op2;	কোন অপারেন্ডের লজিক্যাল NOT অপারেশনের জন্য।

টেবিল : লজিক্যাল অপাটের

কন্ডিশনাল অপারেটর

সি প্রোগ্রামে শর্ত সাপেক্ষে কোনো ভেরিয়েবল বা এক্সপ্রেশনের মান অন্য কোনো ভেরিয়েবল বা এক্সপ্রেশনের মান হিসেবে নির্ধারণ করার জন্য কন্ডিশনাল অপারেটর (?:) ব্যবহৃত হয়। কন্ডিশনাল অপারেটর ব্যবহারের ফরম্যাট হলো-

এখানে স্টেটমেন্ট প্রথমে Test Condition পরীক্ষিত হবে; এই মান সত্য বা অশূন্য হলে Exp1 = Exp2 নির্ধারিত বা সম্পাদিত হবে, আর এই মান মিথ্যা বা শূন্য হলে Exp1 = Exp3 নির্ধারিত বা সম্পাদিত হবে।

ইনক্রিমেন্ট ও ডিক্রিমেন্ট অপারেটর (Increment and Decrement Operator)

ইনক্রিমেন্ট অপারেটর (Increment Operator): প্রোগ্রামে কখনও কখনও ভেরিয়েবলের মান একটা নির্দিষ্ট ইনক্রিমেন্ট বাড়াতে হয়। ভেরিয়েবলের মানকে বর্ধিত (Increment) করার জন্যই মূলত ইনক্রিমেন্ট অপারেটর ব্যবহৃত হয়ে থাকে। এই বর্ধিতকরণ 1 থেকে শুরু করে যে কোনো মান হতে পারে। যেমন: x = x + 1; x = x + 2, x = x + 3 ইত্যাদি। ইনক্রিমেন্টের মান 1 করে বর্ধিত করার জন্য সি এর স্পেশাল ইনক্রিমেন্ট অপারেটর যেমন ++ ব্যবহার করা হয়। এ ধরণের অপারেটর একটিমাত্র অপারেন্ড নিয়ে কাজ করে বলে এদেরকে ইউনারি অপারেটর বলা হয়। যার সাধারণ গঠন (General Syntex) হলো: variable ++ বা ++ variable;

প্রথমটিকে Postfix ইনক্রিমেন্ট আর দ্বিতীয়টিকে Prefix ইনক্রিমেন্ট বলা হয়। সাধারণত while, for ইত্যাদি লুপিং-এর ক্ষেত্রে এ ধরনের ইনক্রিমেন্ট ব্যবহৃত হয়ে থাকে। অবশ্য Postfix ও Prefix-এর সাথে আলাদা আলাদা অর্থ প্রকাশ করে। যেমন, i = 4 এবং j = i++; এ ক্ষেত্রে i এর মান হবে 5, কিন্তু j এর মান হবে 4; আবার যদি i = 4 এবং j = ++i লেখা হয় তাহলে i ও j উভয়ের মান হবে 5 অর্থাৎ, Prefix প্রথমে বাম প্রান্তের variable-এর সাথে 1 যোগ করে, তারপর ফলাফলকে বাম প্রান্তের ভেরিয়েবল দ্বারা অ্যাসাইন করে। অপরদিকে Postfix প্রথমে বাম প্রান্তের variable-এর মান অ্যাসাইন করে, তারপর অপারেন্ডের মান 1 বর্ধিত করে।

অর্থাৎ i++ এর মানে হলো। এর মান 1 বাড়বে, কিন্তু আগের মানটিই ফেরত দিবে। যেমন-

প্রো-৯: ইনক্রিমেন্ট অপারেটরের ব্যবহার

```
#include<stdio
int main()
{
    int j;
    int i=1;
    j = i ++;
    printf("i=%d\n j=%d",i,j);
return 0;
}
```

এখানে, j = i++; লেখার কারণে i এর মান 1 বেড়ে 2 হবে কিন্তু যেহেতু i++ আগের মানটি (1) ফেরত দেবে তাই j এর মান হবে 1। প্রোগ্রামটি লিখে রান করালে আমরা ফলাফল দেখতে পাবো।

Output :

```
i =2
j =1
```

++i এর মানে হলো i এর মান 1 বাড়বে এবং বাড়ার পর নতুন যে মান হবে সে মানটিই ফেরত দিবে। যেমন—

প্রো-১০: ইনক্রিমেন্ট অপারেটরের ব্যবহার

```
#include<stdi
int main()
{
    int j;
    int i=1;
    j = ++i;
    printf("i=%d\n j=%d",i,j);
return 0;
}
```

এখানে, j = ++i লেখার কারণে i এর মান 1 বেড়ে 2 হবে এবং বাড়ার পর নতুন যে মান হবে সে মানটিই ফেরত দিবে। তাই j এর মান হবে 2। নিচের প্রোগ্রামটি লিখে রান করালে আমরা ফলাফল দেখতে পাবো।

Output :

```
i =2
j =2
```

ডিক্রিমেন্ট অপারেটর (Decrement Operator): ইনক্রিমেন্ট অপারেটরের বিপরীতে কাজ করে ডিক্রিমেন্ট অপারেটর। অর্থাৎ ভেরিয়েবলের মানকে হ্রাস (Decrement) করার জন্যই মূলত ডিক্রিমেন্ট অপারেটর ব্যবহৃত হয়ে থাকে। এই হ্রাসকরণ 1 থেকে শুরু করে যে কোন মান হতে পারে। যেমন: x = x-1; x = x-2; x = x-3 ইত্যাদি। ডিক্রিমেন্টের মান 1 করে হ্রাস কারার জন্য সি এর স্পেশাল ডিক্রিমেন্ট অপারেটর ব্যবহার করা হয়, যার সাধারণ গঠন (Syntax) হলো:

variable — অথবা –variable

প্রথমটিকে postfix ডিক্রিমেন্ট এবং দ্বিতীয়টিকে prefix ডিক্রিমেন্ট বলা হয়। সাধারণত for, while লুপিং-এর ক্ষেত্রে এ ধরনের ডিক্রিমেন্ট ব্যবহৃত হয়ে থাকে।

Variable ++ ও ++ variable এর মধ্যে পার্থক্য

variable ++ অথবা i++	++ variable অথবা ++i
১. একে Postfix বলা হয়।	১. একে Prefix বলা হয়।
২. Postfix প্রথমে বাম প্রান্ত variable এর মান অ্যাসাইন করে, তারপর অপারেন্ডের মান 1 বর্ধিত করে।	২. Prefix প্রথমে Operand এর সাথে 1 যোগ করে, তারপর ফলাফলকে বাম প্রান্তে ভেরিয়েবল দ্বারা অ্যাসাইন কর।
৩. উদাহরণ: i = 4 এবং j = i ++; এ ক্ষেত্রে i এর মান হবে 5 কিন্তু j এর মান হবে 4	৩. উদাহরণ: i = 4 এবং j = ++i; এ ক্ষেত্রে i এবং j উভয়ের মান হবে 5

Chapter-4
Input and output operations

After completing this chapter we will learn about-

- What is format specifier
- Input output operations
- What is Backslash character

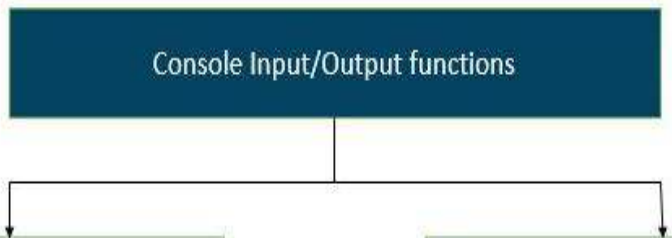
ফরমেট স্পেসিফায়ার

scanf() বা printf() ফাংশন দ্বারা কোনো ডেটা বা ভেরিয়েবলের মান স্টোর বা প্রদর্শনের জন্য নির্দিষ্ট ফরম্যাটের কতগুলো ক্যারেক্টার ব্যবহার করা হয়, যেগুলোকে ফরমেট স্পেসিফায়ার বলা হয়। যেমন %d ফরমেট স্পেসিফায়ারটি integer টাইপের ডেটা ইনপুট বা আউটপুট করার জন্য ব্যবহৃত হয়। নিচের টেবিলে বিভিন্ন ধরনের ফরমেট স্পেসিফায়ারের ব্যবহার উল্লেখ করা হলো:

ফ.স্পেসিফাইয়ার	ব্যবহার	উদাহরণ
%c	single character প্রিন্ট করার জন্য ব্যবহৃত হয়।	scanf("%c" ,&name); printf("%c",name);
%d	integer টাইপের ডেটা ইনপুট বা আউটপুট করার জন্য।	scanf("%d" ,&name); printf("%d",name);
%ld	long integer টাইপের ডেটা ইনপুট বা আউটপুট করার জন্য।	scanf("%ld" ,&name); printf("%ld",name);
%f	floating point টাইপের ডেটা ইনপুট বা আউটপুট করার জন্য।	scanf("%f" ,&name); printf("%f",name);
%lf	double টাইপের ডেটা ইনপুট বা আউটপুট করার জন্য।	scanf("%lf" ,&name); printf("%lf",name);
%u	unsigned integer টাইপের ডেটা ইনপুট বা আউটপুট করার জন্য।	scanf("%u" ,&name); printf("%u",name);
%lu	long unsigned integer ডেটা ইনপুট বা আউটপুট করার জন্য।	scanf("%lu" ,&name); printf("%lu",name);
%o	octal টাইপের ডেটা ইনপুট বা আউটপুট করার জন্য।	scanf("%o" ,&name); printf("%o",name);
%x	hexadecimal integer ডেটা (a,b,.f) ইনপুট/আউটপুট করার জন্য।	scanf("%x" ,&name); printf("%x",name);
%X	hexadecimal integer (A,B,.F) ডেটা ইনপুট/আউটপুট করার জন্য।	scanf("%X" ,&name); printf("%X",name);
%s	string টাইপের ডেটা ইনপুট বা আউটপুট করার জন্য ব্যবহৃত হয়।	scanf("%s" ,&name); printf("%s",name);
%e বা %E	float টাইপ মান %f সায়েন্টিফিক নোটেশনে ইনপুট/আউটপুট।	scanf("%e/%E" ,&name); printf("%c",name);
%g বা %G	float টাইপ মান %f অথবা %e নোটেশনে ইনপুট /আউটপুট করা।	scanf("%g/%G" ,&name); printf("%c",name);
%[^\n]	স্ট্রিং টাইপ মান ইনপুট বা আউটপুট করার জন্য।	scanf("% [^\n]" ,&name); printf("%c",name);
% hd	short integer টাইপ মান ইনপুট বা আউটপুট করার জন্য।	scanf("%hd" ,&name); printf("%c",name);

'সি' প্রোগ্রামিং ভাষায় ইনপুট এবং আউটপুট ফাংশনসমূহঃ

কোন প্রোগ্রামে ডেটা প্রক্রিয়া করার জন্য প্রথমে ডেটা ইনপুট নিতে হয়। প্রোগ্রামে ডেটা ইনপুট নেওয়ার জন্য ব্যবহৃত ফাংশনকে ইনপুট ফাংশন বলে। আবার প্রক্রিয়া পরবর্তী তথ্য আউটপুটে প্রদর্শনের জন্য ব্যবহৃত ফাংশনকে আউটপুট ফাংশন বলে। 'সি' প্রোগ্রামিং ভাষায় ইনপুট নেওয়া এবং আউটপুট দেখানোর জন্য বিভিন্ন লাইব্রেরী ফাংশন রয়েছে। ফাংশনসমূহঃ



Formatted Input/Output function

Type	Input	Output
char	scanf()	printf()
int	scanf()	printf()
float	scanf()	printf()
string	scanf()	printf()

Unformatted Input/Output function

Type	Input	Output
char	getch() getche() getchar()	putch() putchar()
int	-	-
float	-	-
string	gets()	puts()

ফরমেট স্পেসিফায়ারঃ 'সি' প্রোগ্রামের কোন চলকে ফরমেটেড আকারে ডেটা গ্রহণ বা ফরমেটেড আকারে কোন চলকের মান প্রদর্শনের জন্য যথাক্রমে ইনপুট ও আউটপুট ফাংশনে যে সকল ক্যারেক্টার সেট ব্যবহৃত হয় তাদেরে ফরমেট স্পেসিফায়ার বলা হয়। প্রতিটি ফরমেট স্পেসিফায়ার পার্সেন্টেজ ক্যারেক্টার(%) দিয়ে শুরু হয়।

বিভিন্ন ডেটা টাইপের জন্য ফরমেটেড ইনপুট ও আউটপুট ফাংশনে ব্যবহৃত ফরমেট স্পেসিফায়ারসমূহঃ

ফরমেট স্পেসিফায়ার	ব্যবহার	উদাহরণ
%c	char টাইপের ডেটা ইনপুট/আউটপুটের জন্য	scanf("%c",&a); printf("%c",a);
%d	int টাইপের ডেটা ইনপুট/আউটপুটের জন্য	scanf("%d",&a); printf("%d",a);
%f	float টাইপের ডেটা ইনপুট/আউটপুটের জন্য	scanf("%f",&a); printf("%f",a);
%lf	double টাইপের ডেটা ইনপুট/আউটপুটের জন্য	scanf("%lf",&a); printf("%lf",a);
%ld	long int টাইপের ডেটা ইনপুট/আউটপুটের জন্য	scanf("%ld",&a); printf("%ld",a);
%u	unsigned int টাইপের ডেটা ইনপুট/আউটপুটের জন্য	scanf("%u",&a); printf("%u",a);
%o	Octal ডেটা ইনপুট/আউটপুটের জন্য	scanf("%o",&a); printf("%o",a);
%x	Hexadecimal ডেটা ইনপুট/আউটপুটের জন্য	scanf("%x",&a); printf("%x",a);

scanf() ফাংশনের ব্যবহারঃ

পূর্বে ঘোষণাকৃত একটি চলকে ডেটা ইনপুট নেওয়ার জন্য scanf() ফাংশন ব্যবহারের ফরমেটঃ

```
scanf("format_specifier ", &variable_name);
```

উদাহরণঃ

- a চলকে char টাইপের ডেটা ইনপুট নেওয়ার জন্য scanf() ফাংশনঃ scanf("%c", &a);
- a চলকে int টাইপের ডেটা ইনপুট নেওয়ার জন্য scanf() ফাংশনঃ scanf("%d", &a);
- a চলকে float টাইপের ডেটা ইনপুট নেওয়ার জন্য scanf() ফাংশনঃ scanf("%f", &a);
- a চলকে double টাইপের ডেটা ইনপুট নেওয়ার জন্য scanf() ফাংশনঃ scanf("%lf", &a);

printf() ফাংশনের ব্যবহারঃ

printf() ফাংশন দুইভাবে ব্যবহার করা যায়। প্রথমত, কোন কিছু হুবহু আউটপুটে দেখানো। দ্বিতীয়ত, কোন এক বা একাধিক চলকের মান আউটপুটে দেখানো।

কোন কিছু হুবহু আউটপুটে দেখানোর জন্য printf() ফাংশনের ফরমেটঃ

আউটপুটে দেখানোর প্রয়োজনীয় টেক্সটি printf(" "); ফাংশনের ডাবল কোটেশনের মধ্যে লিখতে হয়। যেমন-

```
printf(" Output text should be here ");
```

কোন একটি চলকের মান আউটপুটে দেখানোর জন্য printf() ফাংশনের ফরমেটঃ

```
printf("format_specifier", variable_name);
```

উদাহরণঃ

- a চলকের char টাইপের ডেটা আউটপুটে দেখানোর printf() ফাংশনঃ printf("%c", a);
- a চলকের int টাইপের ডেটা আউটপুটে দেখানোর printf() ফাংশনঃ printf("%d", a);
- a চলকের float টাইপের ডেটা আউটপুটে দেখানোর printf() ফাংশনঃ printf("%f", a);
- a চলকের double টাইপের ডেটা আউটপুটে দেখানোর printf() ফাংশনঃ printf("%lf", a);

উদাহরণঃ

- a চলকের char টাইপের ডেটা আউটপুটে দেখানোর printf() ফাংশনঃ printf("%c", a);
- a চলকের int টাইপের ডেটা আউটপুটে দেখানোর printf() ফাংশনঃ printf("%d", a);
- a চলকের float টাইপের ডেটা আউটপুটে দেখানোর printf() ফাংশনঃ printf("%f", a);
- a চলকের double টাইপের ডেটা আউটপুটে দেখানোর printf() ফাংশনঃ printf("%lf", a);

একাধিক চলকের মান একসাথে আউটপুটে দেখানোর জন্য printf() ফাংশনের ফরমেটঃ

printf("format_specifier1, format_specifier2....", variable_name1, variable_name2...);

একসাথে একাধিক চলকের একই ধরণের ডেটা আউটপুটে দেখানোর printf() ফাংশনের ব্যবহারঃ

- a, b ও c চলকের ডেটা আউটপুটে int টাইপের দেখানোর জন্য printf() ফাংশনঃ
- printf("%d %d %d", a, b, c);

একসাথে একাধিক চলকের ভিন্ন ভিন্ন ধরণের ডেটা আউটপুটে দেখানোর printf() ফাংশনের ব্যবহারঃ

- a,b ও c চলকের ডেটা আউটপুটে যথাক্রমে int, float ও double টাইপের দেখানোর printf() ফাংশনের ব্যবহারঃ
- printf("%d %f %lf", a, b, c);

ব্যাকস্ল্যাশ ক্যারেক্টারঃ বিশেষ কিছু ক্যারেক্টার আছে যেগুলো printf() ফাংশনের মাধ্যমে সরাসরি প্রদর্শন করা যায় না। এই বিশেষ ক্যারেক্টারগুলো প্রদর্শনের জন্য অতিরিক্ত একটি ক্যারেক্টার(\ ) ব্যবহৃত হয় যাকে ব্যাকস্ল্যাশ ক্যারেক্টার বলে।

ব্যাকস্ল্যাশ ক্যারেক্টারের ব্যবহারঃ

ব্যাকস্ল্যাশ ক্যারেক্টার	ব্যবহার	উদাহরণ	আউটপুট
\n	আউটপুট নতুন লাইনে প্রদর্শনের জন্য	printf("Learning\n ICT");	Learning ICT
\t	আউটপুটে horizontal tab প্রদর্শনের জন্য	printf("Learning\t ICT");	Learning ICT
\v	আউটপুটে vertical tab প্রদর্শনের জন্য	printf("Learning\v ICT");	Learning ICT
\a	সতর্ক সংকেত প্রদানের জন্য	printf("\aLearning ICT");	Alerm!
\"	আউটপুটে ডাবল কোটেশন(") প্রদর্শনের জন্য	printf("Learning \"ICT\" ");	Learning "ICT"
\'	আউটপুটে সিঙ্গেল কোটেশন(') প্রদর্শনের জন্য	printf("Learning \'ICT\' ");	Learning 'ICT'
\\	আউটপুটে ব্যাকস্ল্যাশ ক্যারেক্টার (\ ) প্রদর্শনের জন্য	printf("Learning \\ICT\\ ");	Learning \ICT\

ব্যাকস্ল্যাশ ক্যারেক্টারের ব্যবহারঃ

ব্যাকস্ল্যাশ ক্যারেক্টার	ব্যবহার	উদাহরণ	আউটপুট
\n	আউটপুট নতুন লাইনে প্রদর্শনের জন্য	printf("Learning\n ICT");	Learning ICT
\t	আউটপুটে horizontal tab প্রদর্শনের জন্য	printf("Learning\t ICT");	Learning ICT
\v	আউটপুটে vertical tab প্রদর্শনের জন্য	printf("Learning\v ICT");	Learning ICT
\a	সতর্ক সংকেত প্রদানের জন্য	printf("\aLearning ICT");	Alerm!
\"	আউটপুটে ডাবল কোটেশন(") প্রদর্শনের জন্য	printf("Learning \"ICT\" ");	Learning "ICT"
\'	আউটপুটে সিঙ্গেল কোটেশন(') প্রদর্শনের জন্য	printf("Learning \'ICT\' ");	Learning 'ICT'
\\	আউটপুটে ব্যাকস্ল্যাশ ক্যারেক্টার (\ ) প্রদর্শনের জন্য	printf("Learning \\ICT\\ ");	Learning \ICT\
\?	আউটপুটে প্রশ্নবোধক চিহ্ন (?) প্রদর্শনের জন্য	printf(" Are you learning ICT\? ");	Are you learning ICT?


THANK YOU