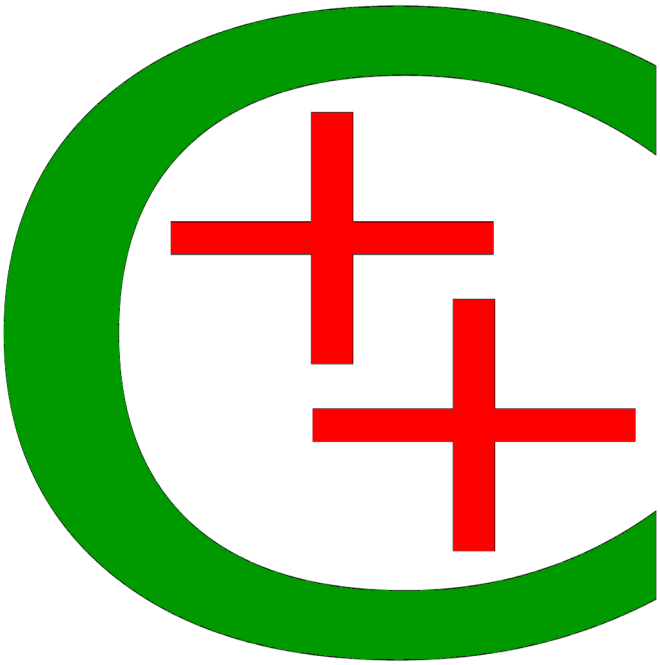
সিপিপি পরিগণনা

c++ programming



নিউটন মু. আ. হাকিম

সূচীপত্র

ফিরিস্তি তালিকা

অধ্যায় ১

ক্রমলেখয়ের কাঠামো (Program Structure)

গণনিতে (computer) নির্বাহযোগ্য (executable) একগুচ্ছ নির্দেশের (instruction) ক্রমকে ক্রমলেখ (program) বলা হয়। আমরা সিপিপি (c++) ভাষায় ক্রমলেখ তৈরী করবো। ক্রমলেখ সাধারণত একটি সম্পাদনা (editor) মন্ত্র (software) ব্যবহার করে তৈরী করা হয়। আমরা একাজে আপাতত ideone.com নামের একটি জালপাতা (webpage) ব্যবহার করবো। সি-পিপি ভাষায় তৈরী ক্রমলেখকে প্রথমে একটি সংকলক (compiler) দিয়ে সংকলন (compile) করে গণনিতে নির্বাহযোগ্য সংকেত (code) তৈরী করা হয়। তারপর সেই সংকেত চালালে (run) বা নির্বাহ (execution) করলে আমরা সাধারণত যন্ত্রালয়ের (console) নজরিতে (monitor) ফলন (output) দেখতে পাই । ক্রমলেখ অনেক সময় আমাদের কাছ থেকে যন্ত্রালয়ের চাপনির (keyboard) বা টিপনির (mouse) মাধ্যমে যোগান (input) নিতে পারে। জেনে রেখো যন্ত্রা-লয় (console) বলতে যোগানের (input) জন্য চাপনি ও টিপনি (keyboard and mouse) আর ফলনের (output) জন্য নজরি (monitor) বুঝানো হয়। ক্রমলেখ লিখতে গেলে যন্ত্রালয় (console) থেকে যোগান (input) নেয়ার ও যন্ত্রালয়ে (console) ফলন (output) দেখানোর কথা তুমি প্রায়শই শুনতে পাবে। কাজেই এগুলো কী বুঝায় সেটা ভালো করে মনে রেখো।

১.১ শুভেচ্ছা বার্তার ক্রমলেখ (Wishing Program)

সিপিপি (c++) ভাষায় এমন একটি ক্রমলেখ (program) রচনা করো যেটি চালালে (run) তোমার ক্রমলেখ ব্যবহারকারীকে শুভেচ্ছা জানাবে। আসলে এটিই হবে সিপিপি ভাষায় তোমার লেখা প্রথম ক্রমলেখ। প্রত্যেক পরিগণনা ভাষায়ই এমন একটা করে ক্রমলেখ রচনা করা হয়।

নীচের ফিরিস্তি ১.১ এ শুভেচ্ছা বার্তা দেখানোর জন্য একটি ক্রমলেখ রচনা করা হয়েছে। আর ক্রমলেখটি সংকলন (compile) করে নির্বাহ (execution) করলে বা চালালে (run) যে ফলন (output) পাওয়া যাবে তাও দেখানো হয়েছে। ওই ক্রমলেখতে মুল যে বিবৃতিটি (statement) আমাদের shuvessa nin দেখাবে সেটি হল `cout << "shuvessa nin" << endl;` এখানে cout হল console out মানে যন্ত্রালয়ের ফলন যন্ত্র (output device)। আর endl হল end line অর্থাৎ যেখানে endl বলা আছে সেখানে ফলনে ওই সারি শেষ হবে। খেয়াল করো আমরা নজরিতে যা দেখাতে চাই তা হুবহু উদ্ধৃতি " " চিহ্নের ভিতরে লেখা হয়েছে। আর << দিয়ে আমরা "shuvessa nin" ও endl কথাগুলোকে cout এর কাছে পাঠাই দেখানোর জন্য।

১.১. শুভেচ্ছা বার্তার ক্রমলেখ (Wishing Program)

স্মরণ রেখো cout এর বিবৃতিটি (statement) ছাড়া আমাদের ক্রমলেখতে আরো অন্যান্য বিবৃতি যেগুলি আছে সেগুলি আমাদের লেখা প্রায় সকল ক্রমলেখতেই থাকবে। আমরা তাই আপাতত ওগুলো একরকম জোর করে মনে রাখার চেষ্টা করবো। তারপরেও অবশ্য আমরা নীচের আলোচনা থেকে সংক্ষেপে জেনে নেব বাঁকী বিবৃতিগুলোর কোনটার কাজ মোটামুটি কী।

ফিরিস্তি ১.১: শুভেচ্ছা জানানোর ক্রমলেখ (Wishing Program)

```
#include <iostream>
using namespace std;
int main()
{
  cout << "shuvessa nin" << endl;

  return 0;
}
```

ফলন (output)

```
shuvessa nin
```

একদম শুরুতে আমরা #include <iostream> ব্যবহার করেছি কারণ iostream নামে একটা শির নথি (header file) আছে যেটা আমরা আমাদের ক্রমলেখতে অন্তর্ভুক্ত করতে চাই। ওই শির নথিতে নানান বিপাতক (function) আছে যেগুলো আমরা পরে জানব ও ব্যবহার করবো। আপাতত জেনে নেই, ওই নথিতে cout আর endl আছে। মুলত আমাদের ক্রমলেখতে cout আর endl ব্যবহার করার জন্যই আমরা iostream অন্তর্ভুক্ত করেছি। এরকম আরো শির নথির (header file) কথা আমরা পরে বিস্তারিত জানবো ও অবশ্যই ব্যবহার করবো।

using namespace std; আমরা ব্যবহার করেছি কারণ cout আর endl আসলে দুটো নাম, আর ওই নাম দুটো সিপিপিতে আগে থেকে বিদ্যমান std (standard বা প্রমিত) নামাধারের (namespace) অন্তর্গত। সিপিপিতে একই নাম ভিন্ন ভিন্ন নামাধারে অন্তর্গত হতে পারে। তো কোনো নাম বললে সেটি কোন নামাধার থেকে আসবে সেটি আমরা আগেই বলে দিচ্ছি, যেমন আমাদের সকল নাম আসলে std নামাধার থেকে আসবে। নামাধার কী তা আর একটু পরিস্কার করে বুঝতে হলে নীচের পরিচ্ছেদের (para) ঢাকার বনাম বগুড়ার গাবতলি নিয়ে আলোচনা পড়ো।

গাবতলি নামে ঢাকায় একটি জায়গা আছে আবার গাবতলি নামে বগুড়ায় আরেকটি জায়গা আছে। তো গাবতলি বলতে গেলে আমাদের বলতে হবে 'বগুড়ার গাবতলি' অথবা 'ঢাকার গাবতলি', কেবল গাবতলি বললে তো বুঝা যাবে না কোথাকার গাবতলি। বিকল্প হিসাবে আমরা আগেই বলে নিতে পারি যে আমরা এখন ঢাকার কথা আলোচনা করছি। তখন কেবল গাবতলি বললেই আমরা বুঝব এটি ঢাকার গাবতলি। আবার যদি আগেই বলে নেই যে এখন থেকে আমরা বগুড়ার কথা আলোচনা করবো তাহলে গাবতলি বললেই আমরা বগুড়ার গাবতলি বুঝব, ঢাকারটা নয়।

উপরের ক্রমলেখতে using namespace std; বলে আমরা আগেই বলে নিয়েছি যে এরপর থেকে আমরা std নামাধার (namespace) নিয়ে কাজ করবো। কাজেই পরে যখন cout আর endl ব্যবহার করেছি, তখন আর std এর কথা বলতে হয় নি। কিন্তু কেউ যদি তার ক্রমলেখতে using namespace std; না লেখে, তাহলে তাকে cout << "shuvessa nin" << endl ; এর বদলে লিখতে হবে std::cout << "shuvessa nin" << std::endl; অর্থাৎ cout আর endl দুটোর পূর্বেই std:: লাগিয়ে নিতে হবে, ঠিক যেমন গাবতলি বলার আগে ঢাকা লাগিয়ে

বলতে হবে ঢাকার গাবতলি। cout আর endl এর আগে std:: না লিখলে ক্রমলেখ সফল ভাবে সংকলন (compile) করা যাবে না, নানান ত্রুটি (error) বার্তা (message) দেখাবে। সংকলন সময়ে দেখানো ত্রুটিবার্তাগুলোকে সংকলন কালীন (compile-time) ত্রুটি বলা হয়।

যে কোন সিপিপি ক্রমলেখতে একটি মুল বিপাতক (function) থাকে main যার নাম। এই main বিপাতকের কোন পরামিতি (parameter) থাকবে না, কাজেই main( ) এর পরে গোল বন্ধনী দুটোর মধ্যে কিছু বলা হয় নি। আর প্রতিটি বিপাতক চাইলে একটি মান ফেরত দেয়, main বিপাতক সাধারণত একটি পূর্ণক (integer) ফেরত দেয়, যা main লেখার আগে int হিসাবে উল্লেখ করা হয়েছে। বিপাতক নিয়ে বিস্তারিত আলোচনা আমরা পরে করবো। আপাতত সংক্ষেপে এইটুকুই জেনে রাখি। তো আমাদের ক্রমলেখতে return 0; বিবৃতিটি আসলে বলছে যে আমাদের main বিপাতকটি শুন্য ফেরত পাঠাবে। কার কাছে ফেরত পাঠাবে? যে আমাদের ক্রমলেখ চালাচ্ছে তার কাছে। main বিপাতক 0 পাঠানো মানে হলো, এটি সফল ভাবে শেষ হয়েছে, কোন ত্রুটি বিচ্যুতি ঘটে নি। 0 ছাড়া অন্যকিছু ফেরত পাঠানো নিয়েও আমরা পরে আলোচনা করবো।

সিপিপিতে দুটো বাঁকা বন্ধনির {} ভিতরে যা থাকে তাকে বলা হয় একটি মহল্লা (block)। প্রতিটি বিপাতকের একটি শরীর (body) থাকে যেটি মহল্লার ভিতরে থাকে। লক্ষ্য করে দেখো আমাদের main বিপাতকের cout আর return দিয়ে শুরু হওয়া বিবৃতি দুটি একটি মহল্লার ভিতরে রয়েছে। আর একটি বিষয় খেয়াল করো, আমাদের বিবৃতিগুলোর শেষে কিন্তু একটি করে দির্তি (semicolon) ; রয়েছে। সিপিপিতে বেশীরভাগ বিবৃতির পরেই আমরা এইরকম দির্তি ; দিয়ে বিবৃতি শেষ করি। ঠিক বাংলা ভাষায় প্রতিটি বাক্যের পরে দাঁড়ি । দেয়ার মতো ব্যাপার।

সব মিলিয়ে এই হল আমাদের প্রথম ক্রমলেখ, যেটা ব্যবহারকারীকে শুভেচ্ছা জানাবে।

১.২ নাম-ধাম-বৃত্তান্তের ক্রমলেখ (Detailing Program)

সিপিপিতে এমন একটি ক্রমলেখ (program) রচনা করো যেটি চালালে ব্যবহারকারীকে তোমার নাম-ধাম-বৃত্তান্ত কয়েক সারিতে মালা (string) আকারে বলে দেয়। সাথে সংখ্যা (number) হিসাবে তোমার বয়স ও তোমার ফলাফলের জিপিএও বলে দেয়।

ফিরিস্তি ১.২: নাম-ধাম-বৃত্তান্তের ক্রমলেখ (Detailing Program)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  cout << "amar nam goni mia" << endl;
  cout << "amar bari bogra" << endl;
  cout << "ami thaki dhakai" << endl;
  cout << "amar boyos " << 20 << " bosor" << endl;
  cout << "amar result gpa " << 3.99 << endl;

  return EXIT_SUCCESS;
}
```

ফলন (output)

```
amar nam goni mia
amar bari bogra
ami thaki dhakai
amar boyos 20 bosor
amar result gpa 3.99
```

উপরের ফিরিস্তি ১.২ এর ক্রমলেখতে আমরা নাম-ধাম-বৃত্তান্ত কয়েক সারিতে দেখিয়েছি। এই ক্রমলেখয়ের cout দিয়ে শুরু বিবৃতিগুলোর সাথে পরের ফলনগুলো মিলিয়ে নাও। লক্ষ্য করো cout দিয়ে "" উদ্ধৃতি অংশগুলোর ভিতরে আমরা যে মালাগুলো (string) দেখাতে বলেছি সেগুলোই ফলনে হুবহু সেভাবেই দেখানো হয়েছে। আর প্রতিবার endl অর্থাৎ end line পেলে ফলন পরের সারিতে চলে গেছে। শেষের দুটো cout বিবৃতিগুলো খেয়াল করো। এইদুটোতে বয়স ও জিপিএ আমরা সংখ্যা হিসাবে দেখিয়েছি। তুমি চাইলে কিন্তু সংখ্যা হিসাবে না দেখিয়ে মালার ভিতরেই দেখাতে পারতে যেমন নীচের মতো, সেক্ষেত্রে ফলন কিন্তু দেখতে একই রকম হতো।

```
cout << "amar boyos 20 bosor" << endl;
cout << "amar result gpa 3.99" << endl;
```

সবশেষে একটা বিষয় খেয়াল করো। আমরা এই ক্রমলেখতে return 0; এর বদলে লিখেছি return EXIT_SUCCESS; আর এই EXIT_SUCCESS আছে cstdlib শির নথিতে (header file)। আমরা তাই #include <cstdlib> লিখে cstdlib শির নথিটিও আমাদের ক্রমলেখতে অন্তর্ভুক্ত করেছি। মনে রাখবে EXIT_SUCCESS এর মান আসলে 0 কিন্তু 0 তো একটা সংখ্যা যেটা দেখে সরাসরি ঠিক অনুধাবন করা যায় না আমরা কী বুঝাতে চাইছি, মানে ক্রমলেখ সফল না বিফল হয়েছে। আমরা তাই স্পষ্ট করে EXIT_SUCCESS লিখবো যাতে চোখে দেখেই আমরা বুঝতে পারি ব্যাপারটা কী। বলে রাখি গণনির (computer) জন্য কিন্তু 0 আর EXIT_SUCCESS একই ব্যাপার কারণ EXIT_SUCCESS এর মান যে 0 ওইটা তো cstdlib নথিতে বলা আছে, সংকলন করার পরে EXIT_SUCCESS আসলে 0 হয়ে যাবে, গণনি ওইটা শুন্যই দেখতে পাবে। আমরা 0 এর বদলে EXIT_SUCCESS আসলে লিখছি কেবল মানুষের বুঝার সুবিধার জন্য, ক্রমলেখ পড়ে চোখে দেখেই যাতে সহজে বুঝা যায় ক্রমলেখটি সফল না বিফল ভাবে শেষ হচ্ছে, সেটাই আমাদের উদ্দেশ্য। তাহলে এখন থেকে ক্রমলেখয়ের main বিপাতকে return 0; না লিখে return EXIT_SUCCESS; লিখবে আর cstdlib শির নথিও অন্তর্ভুক্ত করে নেবে, কেমন!

তো তোমরা এখন থেকে কয়েক সারিতে কিছু দেখানোর ক্রমলেখ রচনা করতে চাইলে এই ক্রমলেখয়ের মতো করে রচনা করবে। দরকার মতো সংখ্যা (number) ও মালা (string) মিশ্রণ করেও কিন্তু যা দেখাতে চাও তা দেখাতে পারবে। চেষ্টা করে দেখো কেমন?

১.৩ ক্রমলেখতে টীকা লিখন (Writing Program Comments)

এমন একটা ক্রমলেখ (program) রচনা করো যেটি বর্তমান সাল ২০১৫ থেকে তোমার বয়স ২০বছর বিয়োগ করে তোমার জন্ম বছর দেখায়। এই ক্রমলেখতে দরকার অনুযায়ী পর্যাপ্ত টীকা (comment) লিখো, যাতে অনেক দিন পরে তুমি যখন ক্রমলেখটি প্রায় ভুলে যাওয়ার মতো অবস্থায় যাবে তখন ক্রমলেখটি আবার দেখতে গিয়ে দ্রুত চোখ বুলিয়েই সহজে বুঝতে পারো যে এটি তোমার কীসের ক্রমলেখ ছিল। ক্রমলেখতে টীকা থাকলে তুমি ছাড়া অন্য কেউও তোমার লেখা ক্রমলেখ পড়ে সহজে বুঝতে পারবে। টীকা লেখা হয় মানুষ যে ভাষায় কথা বলে সেই ভাষায় যেমন

বাংলায় বা ইংরেজীতে, সিপিপি ভাষায়ও নয়, যন্ত্রের ভাষায়ও নয়, কাজেই টীকা লিখলে অনেক দিন পরেও আমাদের ক্রমলেখ বুঝতে সুবিধা হয়।

ফিরিস্তি ১.৩: ক্রমলেখতে টীকা লেখন (Commenting in Programs)

```
// list of header files needed for this program.

#include <iostream>
#include <cstdlib>

using namespace std;  // use the std namespace

int main()
{
  // Subtract 20years from 2015 to get birthyear

  cout << "amar jonmoshal " << 2015 - 20 << endl;

  return EXIT_SUCCESS;  /* return with success */
}
```

ফলন (output)

```
amar jonmoshal 1995
```

উপরের ফিরিস্তি ১.৩ এর ক্রমলেখ খেয়াল করো। কঠিন কিছু নয়। আগের মতোই iostream আর cstdlib অন্তর্ভুক্ত (include) করা আছে। তারপর বলা হয়েছে using namespace std; তারপর মুল বিপাতক (function) হিসাবে int main() যেটির কোন পরামিতি (parameter) নাই কারণ () গোল বন্ধনীর ভিতরে কিছু নাই আর যেটি একটি পূর্ণক (integer) ফেরত দেয় কারণ int বলা আছে শুরুতে। তারপর মুল বিপাতকের শরীরে দুটো {} বাঁকাবন্ধনীর ভিতরের মহল্লায় (block) বলা আছে cout << "amar jonomoshal " << 2015 - 20 << endl; অর্থাৎ ফলনে amar jonomoshal দেখিয়ে তারপর 2015 থেকে 20 বিয়োগ করলে যে 1995 পাওয়া যায় তা দেখাবে। তারপর মহল্লার ভিতরে শেষ বিবৃতি (statement) আছে return EXIT_SUCCESS; যা আগের মতোই বলছে যে আমাদের ক্রমলেখ ওইখানে সফল ভাবে শেষে হয়ে বের হয়ে যাবে। EXIT_SUCCESS নিয়ে আমরা পাঠ ১.২ এ বিস্তারিত আলোচনা করেছি, ওই পাঠ থেকেই দেখে নিতে পারো, কাজেই সেটা আবার এখানে আলোচনা করছি না।

যাইহোক, খেয়াল করে দেখো ওপরে বর্ণিত বিষয়গুলো ছাড়াও উপরের ক্রমলেখতে আরো কিছু বাক্য ও সারি দেখা যাচ্ছে যেমন প্রথম সারিটিই হল // list of header files needed for this ।program এই বাক্যটি আসলে আমাদের ক্রমলেখয়ের অংশ নয়, অর্থ্যাৎ ক্রমলেখ যখন চালানো (run) হবে তখন এই বাক্যের কোন প্রভাব থাকবে না। ক্রমলেখ এমন ভাবে চলতে থাকবে যাতে মনে হবে ওই বাক্যটি যেন ওখানে নাই। এরকমের বাক্যগুলোকে বলা হয় টীকা (comment)। খেয়াল করো টীকার বাক্যটির একদম সামনে রয়েছে // অর্থাৎ সামনের দিকে হেলানো দুটো দাগ। ওই দুটো দাগ হতে শুরু করে ওই সারিতে তারপরে যাই থাকবে সব মিলিয়ে হবে একটি টীকা। এইরকম টীকা যেহেতু কেবল এক সারিতে সীমাবদ্ধ তাই একে বলা হয় সারি টীকা (line comment)। সিপিপি ভাষায় অধিকাংশ সময়ই সারি টীকা ব্যবহার করা হয়।

সারি টীকা যদি সারির একদম শুরুতে লেখা হয় তাহলে সাধারণত এটি টীকার ঠিক নীচে যে সংকেত (code) থাকে তার জন্য লেখা হয়। যেমন // list of header files needed for this program এই টীকাটি একদম সারির শুরু থেকে লেখা হয়েছে, এটি তাই পরের দুই সারিতে #include <iostream> আর #include <cstdlib> কেন লেখা হয়েছে সেটি ব্যাখ্যা করছে। সারি টীকা অনেক সময় সারির শেষ দিকেও লেখা হয়। যেমন // we will use the std namespace টীকাটি লেখা হয়েছে using namespace std; দিয়ে শুরু হওয়া সারির শেষে। সারির শেষ দিকে লেখা এইরকম সারি টীকা সাধারণত সারির প্রথমে যে সংকেত (code) লেখা হয়েছে তা ব্যাখ্যা করতে ব্যবহার করা হয়। অনেক সময় টীকা লিখা হয় শুরুতে /* আর শেষে */ চিহ্ন দিয়ে, যেমন return EXIT_SUCCESS; এর সারিতে শেষে লেখা হয়েছে। এইরকম টীকা একাধিক সারি মিলিয়ে হতে পারে, তাই এদেরকে সারি টীকা না বলে মহল্লা টীকা (block comment) বলা হয়। সিপিপিতে আমরা অধিকাংশ সময় আসলে সারি টীকাই ব্যবহার করি।

তুমি যখন তোমার ক্রমলেখতে টীকা লিখবে তখন হয়তো ইংরেজীতেই টীকা লিখবে। অথবা ইংরেজী অক্ষরে বাংলায়ও টীকা লিখতে পারো। আজকাল অনেক সংকলক (comiler) ও সম্পাদক (editor) ইউনিকোড (unicode) সংকেত বুঝতে পারে। কাজেই টীকা বাংলায়ও লেখা সম্ভব। আমরা এরপর থেকে সিপিপিতে লেখা সকল ক্রমলেখতে টীকা বাংলায় লিখবো, যাতে আমরা আমাদের নিজের ভাষায় সহজে বুঝতে পারি। এগুলো যেহেতু নির্বাহ execution) হবে না, কাজেই খামোকা কেন কষ্ট করে ইংরেজীতে লিখতে যাবো! আর বিদেশী কেউ তো আমাদের ক্রমলেখয়ের সংকেত দেখবে না, কাজেই আমরা আমাদের বাংলা ভাষাতেই টীকা লিখবো। তবে মনে রাখবে বিদেশী কারো পড়ার সম্ভাবনা থাকলে আমাদের টীকা সহ সবকিছু ইংরেজী ভাষাতেই লিখতে হবে। তাহলে সারি টীকা আর মহল্লা টীকা শেখা হলো। এখন থেকে ক্রমলেখ লেখার সময় যথেষ্ট পরিমানে টীকা দিবে কেমন? আমিও ক্রমলেখগুলোতে টীকা দেবো, যাতে তোমাদের বুঝতে সুবিধা হয়।

১.৪ ক্রমলেখতে ফাঁকা দেওয়া (Spacing and Indentation)

সিপিপি ক্রমলেখ (program) লিখতে কখন নতুন সারি শুরু করবে? কখন ফাঁকা ফাঁকা করে লিখবে? কখন সারিতে একটু ছাড়ন দিয়ে লিখবে। একটি ক্রমলেখ লিখে এই বিষয়গুলো আলোচনা করো। চলো আমরা আমাদের শুভেচ্ছা জানানোর ছোট ক্রমলেখটি দিয়েই আলোচনা করি।

ফিরিস্তি ১.৪: ক্রমলেখতে ফাঁকা দেওয়া (Spacing in Programs)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  cout << "shuvessa nin" << endl;

  return EXIT_SUCCESS;
}
```

উপরের ফিরিস্তি ১.৪ এ ক্রমলেখতে আমরা আপাতত টীকা (comment) লিখি নাই। এই ক্রমলেখতে # বা কাটাকাটি (octothorpe) চিহ্ন দিয়ে শুরু হওয়া সারিগুলো তোমাকে আলাদা

আলাদা সারিতে লিখতে হবে। আর এটি সারির শুরু থেকে হলেই ভালো। তুমি অবশ্য সারির শেষের দিকে চাইলে সারি টীকা (line comment) লিখতে পারো যাতে বুঝা যায় ওই সারির শুরুর দিকে তুমি আসলে কী করতে চেয়েছো। তোমার ক্রমলেখতে লেখা কাটাকাটি চিহ্ন # দিয়ে শুরু হওয়া সারিগুলো সাধারণত সংকলক (compiler) দিয়ে প্রক্রিয়া করা হয় না। আলাদা একটি মন্ত্র (software) যার নাম পুর্ব-প্রক্রিয়ক (preprocessor) সেটা দিয়ে সংকলন করারও আগে এইগুলো প্রক্রিয়া করা হয়, কাজটা বেশীর ভাগ সময়ে অবশ্য সংকলকই করিয়ে নেয়। পূর্ব-প্রক্রিয়ক (preprocessor) বিষয়ে বিস্তারিত আলোচনা আমরা পরে করবো।

```
#include <iostream> //  যোগান ফলন স্রোত শির নথি অন্তর্ভুক্ত হল
```

কোন বিবৃতি পুর্বপ্রক্রিয়ক (preprocessor) না সংকলক (compiler) দিয়ে প্রক্রিয়াকরণ হবে এটা বুঝার আরেকটা সহজ উপায় আছে। এইরকম দির্তি ; (semicolon) আর মহল্লার (block) জন্য যে বাঁকা বন্ধনী } ব্যবহৃত হয় তা দিয়ে শেষ হওয়া বিবৃতিগুলো সাধারণত সংকলক দিয়ে প্রক্রিয়াকরন করা হবে, পূর্ব-প্রক্রিয়ক দিয়ে নয়। যাইহোক, সংকলক দিয়ে যে সংকেতগুলো (code) প্রক্রিয়া করা হয় সেগুলো যে ভিন্ন ভিন্ন সারিতেই লিখতে হবে, বা অনেক ফাঁকা ফাঁকা (space) করেই লিখতে হবে এ রকম কোন কথা নেই। তুমি চাইলে তোমার পুরো ক্রমলেখতে থাকা সকল সংকলনযোগ্য সংকেত এক সারিতে লিখতে পারো। যেমন উপরের ক্রমলেখয়ের সংকলনযোগ্য অংশটুকু আমরা চাইলে নীচের মতো করে টানা এক সারিতে লিখতে পারি।

```
#include <iostream>
#include <cstdlib>
using namespace std; int main() { cout << "shuvessa
 nin" << endl; return EXIT_SUCCESS; }
```

উপরে যদিও দুই সারিতে দেখা যাচ্ছে আমরা আসলে using থেকে শুরু করে } পর্যন্ত টানা একসাথে লিখেছি, কিন্তু এখানে পাশের দিকে স্থানের স্বল্পতার কারণে টানা সারিটি ভেঙে দুই সারি হয়ে গেছে। তোমার সম্পাদকে (editor) এ যদি পাশের দিকে অনেক জায়গা থাকে তুমি এক সারিতেই লিখতে পারবে। আসলে নুন্যতম একটি ফাঁকা (space) দেয়া বাধ্যতামুলক হয়ে যায় যখন পরপর দুটো শব্দ লেখা হয়। যেমন using, namespace, std, int, main এইরকম শব্দ পরপর দুটো থাকলে তোমাকে কমপক্ষে একটি ফাঁকা (space) দিতে হবে। দুটো চিহ্ন যেমন বন্ধনী ( ) বা দির্তি ; বা আরো অনেক প্রতীক আছে, এইগুলো পরপর দুটো থাকলেও কোন সমস্যা নাই। অর্থ্যাৎ একাধিক প্রতীক কোন ফাঁকা না দিয়েও তুমি একসাথে লিখতে পারবে।

এখন প্রশ্ন করতে পারো ফাঁকা দেয়া যদি ব্যাপার না হয়, তাহলে ক্রমলেখ লিখতে কেন ফাঁকা দেবো। বেশী বেশী ফাঁকা আসলে গণনির (computer) জন্য দরকার নেই কিন্তু দরকার মানুষের জন্য। আগের পাঠের কথা মনে করো। আমরা কেন টীকা (comment) লিখেছিলাম? টীকা তো আর নির্বাহিত হয় না। আমরা যাতে অনেকদিন পরে ক্রমলেখয়ের সংকেত (code) দেখে সহজে বুঝতে পারি, আমরা তাই টীকা লিখেছিলাম। তো ক্রমলেখ যদি পুরোটা একটা লম্বা সারি হয়, আমাদের মানুষের পক্ষে সেটা দেখে বুঝে ওঠা খুবই কষ্টকর হবে। মুলত আমাদের মানুষের বুঝার সুবিধার্থে আমরা ক্রমলেখ সারিতে সারিতে ভেঙ্গে ভেঙ্গে লিখি বা দরকার মতো একসাথে লিখি।

ক্রমলেখতে ফাঁকা দেয়ার ব্যাপারটি বাংলায় বা ইংরেজীতে রচনা লেখার মতোই, কখন তুমি আলাদা বাক্য করবে, কখন তুমি আলাদা পরিচ্ছেদ (para) করবে, কখন তুমি আলাদা অনুচ্ছেদ (section) করবে, এই রকম। কোন বিষয়ের সাথে বেশী সম্পর্কিত বিবৃতিগুলো আমরা সাধারণত পরপর সারিতে কোন ফাঁকা (blank line) না দিয়ে লিখবো। আর দুটো বিষয়ের সারিগুলোর মাঝে হয়তো এক সারি ফাঁকা দিয়ে লিখবো, আর বিষয়গুলোর মধ্যে খুব বেশী যোগাযোগ না থাকলে হয়তো আমরা দুই বা আরো বেশী সারি ফাঁকা দিয়ে লিখবো। তাহলে এখন থেকে ক্রমলেখ লেখার

সময় দরকার মতো ফাঁকা দিয়ে দিয়ে লিখবে যাতে তোমার ক্রমলেখ পড়া সহজ হয়।

সবচেয়ে উপরে যেভাবে আমরা ক্রমলেখ লিখেছি সেখানে আরো একটা ব্যাপার খেয়াল করো, আমরা cout বা return এর বিবৃতিগুলো লেখার আগে তাদের নিজ নিজ সারিতে বেশ কিছুটা ফাঁকা দিয়ে লিখেছি, একদম সারির শুরু থেকে লিখি নাই। এটি কেন করলাম? এটি করলাম এ কারণে যে ওই দুটো সারি আসলে আমাদের মহল্লার ভিতরে আছে। লক্ষ্য করো মহল্লার বাঁকা বন্ধনী দুটো কেমন দেখেই বুঝা যায় যে এরা দুজনে দুজনার। আর মহল্লার ভিতরের বিবৃতিদুটো কেমন একটু ভিতরের দিকে থাকায় পরিস্কার বুঝা যায় যে ওরা আসলেই ওই মহল্লার ভিতরে। তো দরকার মতো কোন বিবৃতি এরকম সারির একটু ভিতরের দিকে থেকে লেখার ব্যাপারটিকে বলা হয় ছাড়ন দেয়া (indentation)। ক্রমলেখ লেখার সময় এখন থেকে তোমরা অবশ্যই দরকার মতো ছাড়ন দিয়ে লিখবে, তাহলে দেখবে ক্রমলেখ পড়া ও বোঝা কত সহজ হয়ে যায়।

এই পর্যায়ে জিজ্ঞেস করতে পারো, প্রত্যেক সারিতে এভাবে অতগুলা করে ফাঁকা চাপবো কেমনে এইটা তো বিরক্তিকর। আসলে তোমার চাপনিমাঁচায় (keyboard) একটা লম্ফ (tab) চাপনি আছে, দেখো ওইটা চাপলে একসাথে ৪টা বা ৮টা ফাঁকা (space) এর সমপরিমান ফাঁকা একবারে আসে। তো দরকার মতো একবার বা দুবার লম্ফ চাপলেই হয়ে গেলো। কাজেই ক্রমলেখ লেখার সময় কখনোই এই আলসেমি টুকু করবে না। ছাড়ন দেয়া ক্রমলেখ লেখার জন্য গুরুত্বপূর্ণ ব্যাপার, সুন্দর দেখা যাওয়া আর তাড়াতাড়ি পড়ার জন্য দরকারী, ক্রমলেখতে কোন ভুল থাকলে আমরা যখন ভুল বের করতে চাই তখনও খুব খুব দরকারী, বড় বড় ক্রমলেখ যখন লিখবে তখন ব্যাপারটা খানিকটা ঠেকে ঠেকে শিখে অভিজ্ঞতা দিয়ে ভালো করে বুঝতে পারবে।

১.৫ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. ক্রমলেখ (program) বলতে কী বুঝো? ক্রমলেখ কি কেবল গণনিতেই নির্বাহযোগ্য?

২. সিপিপি ভাষায় ক্রমলেখ তৈরী হতে সেটার ফলাফল দেখা পর্যন্ত কী কী ঘটনা ঘটে?

৩. যন্ত্রালয় (console) কী? এর যোগান (input) ও ফলন (output) যন্ত্রগুলো কী কী?

৪. সিপিপিতে শিরনথি (header file) বলতে কী বুঝো? আমাদের ক্রমলেখগুলোতে শির নথি iostream ও cstdlib আমরা কেন ব্যবহার করেছি?

৫. নামাধার (namespace) কী? বাস্তব জীবনে ও পরিগণনায় উদাহরণ সহ ব্যাখ্যা করো।

৬. সিপিপিতে main বিপাতক হতে ফেরতের সময় return 0; না লিখে তার বদলে return EXIT_SUCCESS; লিখা কেন উত্তম? ব্যাখ্যা করো।

৭. ক্রমলেখতে ছাড়ন দেয়া (indentation) মানে কী? ছাড়ন দেয়ার পক্ষে-বিপক্ষে যুক্তি লিখ। ক্রমলেখ কেন বেশ ফাঁকা ফাঁকা করে লিখা উচিৎ?

৮. ক্রমলেখতে টীকা (comment) লেখা কী? ক্রমলেখতে টীকা (comment) লিখার কয়েকটি কারণ ব্যাখ্যা করো? সারি (line) টীকা ও মহল্লা (block) টীকা কী?

৯. একটি সিপিপি ক্রমলেখতে (program) নীচের কোন বিপাতকটি অবশ্যই থাকতে হবে?

ক) start() খ) system() গ) main() ঘ) program()

১০. ক্রমলেখ সফল ভাবে শেষ হলে main বিপাতক হতে সাধারণত কত ফেরত পাঠানো হয়?

ক) -1 খ) 0 গ) 1 ঘ) কিছুই না

১১. সিপিপিতে মহল্লা (block) বুঝানোর জন্য নীচের কোনগুলো ব্যবহার করা হয়?

ক) { } খ) < > গ) ( ) ঘ) begin end

১২. সিপিপিতে একটি বিবৃতির (statement) শেষে সাধারণত কোন চিহ্ন ব্যবহার করা হয়?

ক) . খ) ; গ) : ঘ) ,

১৩. সিপিপিতে নীচের কোনটি সঠিক টীকা (comment)?

ক) */ টীকা */ খ) ** টীকা ** গ) /* টীকা */ ঘ) { টীকা }

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. নীচের কথাগুলো ফলনে (output) দেখানোর জন্য সিপিপিতে একটি ক্রমলেখ লিখো। দেখতে সুন্দর লাগার জন্য তোমার ক্রমলেখতে দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো টীকা (comment) লিখবে।

```
tomar boyosh noy bosor.
porigonona shikhte chao?
porigonona ki sohoj na!
```

২. সিপিপিতে একটি ক্রমলেখ রচনা করো যেটি নীচের নকশাটির মতো নকশা তৈরী করে। খেয়াল করে দেখো নকশাটি বাংলা অঙ্ক 8 এর মতো। তুমি চাইলে আরো নানান নকশা, নানান বর্ণ বা অঙ্ক নিজের মতো করে ভেবে নিয়ে সেইমতো নকশা তৈরী করতে পারো। যাইহোক দেখতে সুন্দর লাগার জন্য তোমার ক্রমলেখতে দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো টীকা (comment) লিখবে।

```
*****
*   *
* * *
*   *
*****
```

১.৫. অনুশীলনী সমস্যা (Exercise Problems)

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন যাতে একটু সাহায্য কেবল পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. নীচের কথাগুলো ফলনে (output) দেখানোর জন্য সিপিপিতে একটি ক্রমলেখ লিখো। দেখতে সুন্দর লাগার জন্য তোমার ক্রমলেখতে দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো টীকা (comment) লিখবে।

```
tomar boyosh noy bosor.
porigonona shikhte chao?
porigonona ki sohoj na!
```

ফিরিস্তি ১.৫: অণুপ্রেরণার ক্রমলেখ (Inspiring Programming)

```
#include <iostream> //  cout ব্যবহার করার জন্য
#include <cstdlib>  //  EXIT_SUCCESS এর জন্য

using namespace std;// প্রমিত নামাধার ব্যবহারের জন্য

int main()
{
  //  দরকারী কথাগুলো ফলনে দেখাও
  cout << "tomar boyosh noy bosor." << endl;
  cout << "porigonona shikhte chao?" << endl;
  cout << "porigonona ki sohoj na!" << endl;

  return EXIT_SUCCESS;  //  সফল সমাপ্তি
}
```

২. সিপিপিতে একটি ক্রমলেখ রচনা করো যেটি নীচের নকশার মতো নকশা তৈরী করে। খেয়াল করে দেখো নকশাটি বাংলা অঙ্ক ৪ এর মতো। তুমি চাইলে আরো নানান নকশা, নানান বর্ণ বা অঙ্ক নিজের মতো করে ভেবে নিয়ে সেইমতো নকশা তৈরী করতে পারো। যাইহোক দেখতে সুন্দর লাগার জন্য তোমার ক্রমলেখতে দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো টীকা (comment) লিখবে।

```
*****
*   *
* * *
*   *
*****
```

এই ক্রমলেখটি কিন্তু অনেক মজার তাই না। তুমি কি বর্ণমালার প্রতিটা বর্ণ আর ০-৯ দশটা অঙ্কের জন্যেই এরকম নকশা তৈরী করতে পারবে? রাস্তাঘাটে বা বিয়ে বাড়িতে অনেক সময় ছোট ছোট বাতি দিয়ে নানান কিছু লেখা হয়, আসলে এই নকশাগুলোর মতো করে নকশা বানিয়েই সেগুলো করা হয়। গণনিতে (computer) এর নজরির (monitor) পর্দায়ও

অনেক কিছু এভাবে দেখানো হয়। আসলে যে কোন ছবিই এরকম অসংখ্য বিন্দুর সমন্বয়ে তৈরী, কিছু বিন্দু জ্বালানো, কিছু বিন্দু নেভানো। যে বিন্দুগুলো জ্বালানো সেগুলো হলো * আর যেগুলো নেভানো সেগুলো ফাঁকা। তো চলো আমরা ক্রমলেখটি দেখি।

ফিরিস্তি ১.৬: নকশা আঁকার ক্রমলেখ (Program Drawing Designs)

```
#include <iostream> //  cout ব্যবহার করার জন্য
#include <cstdlib>  //  EXIT_SUCCESS এর জন্য

using namespace std;// প্রমিত নামাধার ব্যবহারের জন্য

int main()
{
  // দরকার মতো * ও ফাঁকা দিয়ে নকশা
  cout << "*****" << endl;
  cout << "*   *" << endl;
  cout << "* * *" << endl;
  cout << "*   *" << endl;
  cout << "*****" << endl;

  return EXIT_SUCCESS;  //  সফল সমাপ্তি
}
```

১.৬ গণনা পরিভাষা (Computing Terminologies)

- কাটাকাটি (octothorpe) #
- ক্রমলেখ (program)
- গণনি (computer)
- চাপনি (keyboard)
- চালানো (run)
- ছাড়ন দেয়া (indentation)
- জালপাতা (webpage)
- টিপনি (mouse)
- টীকা (comment)
- ত্রুটি (error)
- দির্তি (semicolon) ;
- নজরি (monitor)
- নামাধারের (namespace)
- নির্দেশ (instruction)
- নির্বাহ (execution)
- নির্বাহযোগ্য (executable)
- পরামিতি (parameter)
- পরিচ্ছেদের (para)
- পুর্ব-প্রক্রিয়ক (preprocessor)
- পূর্ণক (integer)
- প্রমিত (standard)
- ফলন (output)

১.৬. গণনা পরিভাষা (Computing Terminologies)

- ফলন যন্ত্র (output device)
- বার্তা (message)
- বিপাতক (function)
- বিবৃতি (statement)
- মন্ত্র (software)
- মহল্লা (block)
- মহল্লা টীকা (block comment)
- মালা (string)
- যন্ত্রালয় (console)
- যোগান (input)
- শরীর (body)
- শির নথি (header file)
- সংকলক (compiler)
- সংকলন (compile)
- সংকলন কালীন (compile-time)
- সংকেত (code)
- সংখ্যা (number)
- সম্পাদনা (editor)
- সারি টীকা (line comment)

অধ্যায় ২

চলক ও ধ্রুবক
(Variables and Constants)

চলকের (variable) মান (value) বদলানো যায় কিন্তু ধ্রুবকের (constant) মান বদলানো যায় না। ক্রমলেখতে উপাত্ত (data) সরাসরি (directly) না লিখে চলক বা ধ্রুবকের মাধ্যমে ব্যবহার করলে একরকমের পরোক্ষতা (indirection) তৈরী হয়। ফলে উপাত্ত ঠিক কতো সেটা না ভেবে উপাত্তটি কীসের আর তার প্রক্রিয়াকরণ কেমন সেটা ভেবে ক্রমলেখ তৈরী সহজ হয়ে যায়।

২.১ চলকের ব্যবহার (Using Variables)

একটি আয়তের দৈর্ঘ্য ৫ মিটার, প্রস্থ ৩ মিটার। সিপিপি ভাষায় এইরূপ আয়তের ক্ষেত্রফল ও পরিসীমা বের করার ক্রমলেখ (program) রচনা করো। এই ক্রমলেখতে তোমাকে চলক (variable) ব্যবহার করতে হবে, সরাসরি সুত্র থেকে ফলন (output) দেয়া যাবে না।

আমরা আগে এই সমস্যার জন্য সংক্ষিপ্ত ক্রমলেখটা দেখি যেটাতে চলক ব্যবহার না করে একদম সরাসরি সুত্র ব্যবহার করে ক্ষেত্রফল ফলনে (output) দেখানো হবে। আমরা জানি দৈর্ঘ্য আর প্রস্থের গুণফল হল ক্ষেত্রফল আর দৈর্ঘ্য ও প্রস্থের যোগফলের দ্বিগুণ হলো পরিসীমা।

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  cout << "khetrofol holo " << 5 * 3
       << " borgometer" << endl;
  cout << "porishima holo " << 2*(5+3)
       << " meter" << endl;

  return EXIT_SUCCESS;
}
```

২.১. চলকের ব্যবহার (Using Variables)

উপরে আমরা যে ক্রমলেখ লিখলাম আমরা কিন্তু ওইটা চাই না। ওইখানে সংখ্যাগুলো সরাসরি সুত্রে বসিয়ে হিসাব করে ফলন (output) দেখানো হয়েছে। আমরা চাই ক্ষেত্রফল আর পরিসীমার সুত্রগুলো চলকের নাম দিয়ে লিখতে আর সুত্র লিখার আগে চলকগুলোর মান দিয়ে দিতে। চলক ব্যবহারের নানান সুবিধা আছে। যেমন একটি সুবিধা হলো সুত্রে চলকের নাম থাকায় সুত্র দেখেই সহজে বুঝা যায় কীসের সুত্র, যেমন ফিরিস্তি ২.১ দেখো। আর একটি সুবিধা হলো কেউ যদি বলে ৫ না দৈর্ঘ্য হবে ৬, উপরের ক্রমলেখতে কিন্তু দুইখানে 5 বদলাইয়া 6 করতে হবে। ছোট একটা ক্রমলেখতেই যদি দুইখানে বদলাতে হয়, তাহলে বড় একটি ক্রমলেখয়ের কথা চিন্তা করো, সেটাতে আরো কত জায়গায় যে বদলাতে হবে ইয়ত্তা নাই। আমরা এ কারণে চলক ব্যবহার করবো।

ফিরিস্তি ২.১: ক্রমলেখতে চলকের ব্যবহার (Variables in Programs)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int doirgho;  // এই চলকে থাকবে বর্গের দৈর্ঘ্য।
  int prostho;  // এই চলকে থাকবে বর্গের প্রস্থ।
  int khetrofol;  // এই চলকে থাকবে বর্গের ক্ষেত্রফল।
  int porishima;  // এই চলকে থাকবে বর্গের পরিসীমা।

  doirgho = 5;  // দৈর্ঘ্যের এই মান বলে দেয়া আছে
  prostho = 3;  // প্রস্থের এই মান বলে দেয়া আছে।

  // ক্ষেত্রফল বের করার সুত্র হল দৈর্ঘ্য আর প্রস্থের গুণফল।
   khetrofol = doirgho * prostho;

  // পরিসীমা বের করার সুত্র হল দৈর্ঘ্য ও প্রস্থের যোগফলের দ্বিগুন
  porishima = 2*(doirgho + prostho);

  // এবার ক্ষেত্রফল আর পরিসীমা ফলন দেয়া হবে
  cout<< "khetrofol holo " << khetrofol
      << " borgometer" << endl;
  cout << "porishima holo " << porishima
      << " meter" << endl;

  return EXIT_SUCCESS;
}
```

ফলন (output)

```
khetrofol holo 15 borgometer
porishima holo 16 meter
```

উপরের ফিরিস্তি ২.১ এর ক্রমলেখতে খেয়াল করো আমরা দৈর্ঘ্য, প্রস্থ, ক্ষেত্রফল, আর পরিসীমার জন্য চারটা চলক নিয়েছি যাদের নাম হলো doirgho, prostho, khetrofol, porishima। তুমি কিন্তু চাইলে এই নামগুলো ইংরেজী শব্দেও দিতে পারতে যেমন length, width, area, perimeter। আবার তুমি চাইলে শব্দগুলোর প্রথম অক্ষর নিয়ে এক অক্ষরের নামও দিতে পারতে যেমন l, w, a, p। তবে আমরা সবসময় চাই এমন নাম দিতে যাতে নামগুলো দেখলেই বুঝা যায় ওই চলকটা কী কাজে ব্যবহার হবে। এক অক্ষরের নাম দিলে অনেক সময় বুঝা যায় কিন্তু একই অক্ষর দিয়ে যদি একাধিক চলকের নাম শুরু হয়, তাহলে মুশকিল হয়ে যায়। অনেকে আবার খালি x, y, z, অথবা a, b, c এই রকম নাম দেয়। ওই রকম নাম দিলে পরে ক্রমলেখ বুঝতে তোমার নিজের বা অন্য কেউ যে পড়বে তার খুবই সমস্যা হবে। সময় নষ্ট করে বের করতে হবে কোন চলক আসলে কী কাজে ব্যবহার করা হয়েছে। কাজেই সবসময় অর্থবোধক আর যথেষ্ট বড় নাম দিতে চেষ্টা করবে, যাতে নাম দেখেই তার উদ্দেশ্য বুঝা যায়। সিপিপিতে চলকের অর্থবোধক (semantic) ও গঠনসিদ্ধ (syntax) নাম দেয়ার বিষয়ে আমরা পরের কোন পাঠে বিস্তারিত আলোচনা করব।

এখন একটা বিষয় খেয়াল করো আমরা এখানে চলকগুলোর নামের আগে লিখেছি int যেটা আসলে integer এর সংক্ষিপ্ত। integer হল পূর্ণক বা পূর্ণ সংখ্যা। আমরা চলকের নামের আগে এই রকম int লিখে বুঝিয়েছি যে আমাদের এই চলকগুলোর মান হবে পূর্ণক, আমরা কোন ভগ্নাংশ ব্যবহার করবো না। তুমি যদি ভগ্নাংশ ব্যবহার করতে চাও তাহলে তোমাকে int এর বদলে float লিখতে হবে। float হল একরকমের ভগ্নাংশ। আমরা সেই আলোচনা পরে আরো বিস্তারিত করবো। তবে int এর বদলে float লিখলে আমাদের ক্রমলেখতে কিন্তু আর কোথাও কোন কিছু বদলাতে হবে না, ঠিক কাজ করবে। আমরা আপাতত int রেখেই এই পাঠের আলোচনা চালাই।

তো উপরের ক্রমলেখতে আমরা যখন লিখলাম int doirgho; এর মানে হলো doirgho নামের আমাদের একটা চলক আছে আর তার মান হবে পূর্ণক। এইযে int doirgho; লিখে এই বিষয়গুলো বুঝাইলাম এটাকে বলা হয় চলক ঘোষনা (variable declaration)। চলক ঘোষনা করলে তারপর থেকেই চলকটি পরবর্তী যেকোন বিবৃতিতে (statement) ব্যবহার করা যায়, কিন্তু ঘোষনা করার সাথে সাথে ওইখানে চলকের মান কত সেইটা কিন্তু আমরা জানিনা, সাধারনত চলকে তখন একটা উল্টাপাল্টা মান থাকে। এইটা নিয়ে আমরা পরে আরো আলোচনা করবো। এই ক্রমলেখতে আমরা দেখছি এর পরে doirgho = 5; লিখে অর্থাৎ = চিহ্ন ব্যবহার করে আমরা doirgho চলকের মান আরোপ (value assign) করেছি 5। সুতরাং এরপর থেকে doirgho চলকের মান হবে 5। একই ভাবে prostho চলকের মানও আমরা 3 আরোপ করেছি।

এবার খেয়াল করো, চলকের মান আরোপণ শেষ হলে আমরা ক্ষেত্রফল আর পরিসীমার সুত্রগুলো লিখেছি, সেখানে কিন্তু এবার মানগুলো সরাসরি লিখি নাই, তার বদলে চলকগুলো ব্যবহার করেছি। এইখানে হিসাব করার সময় চলকের যে মান থাকবে সেইটাই আসলে ব্যবহার হবে। উপরে যদি doirgho চলকের মান থাকে 5 তাহলে 5 ধরে হিসাব হবে, আর যদি পরে doirgho এর মান 5 এর বদলে 6 করে দেয়া হয়, তাহলে 6 ব্যবহার হবে। এই পরিবর্তন কেবল মান আরোপণের ওইখানে করলেই কাজ হয়ে যাবে, সারা ক্রমলেখতে করতে হবে না। তবে একটা গুরুত্বপূর্ণ বিষয় বলি এখানে doirgho আর prostho চলক দুটিতে মান আরোপণ কিন্তু ক্ষেত্রফল আর পরিসীমার সুত্রের ব্যবহারের আগেই করতে হবে। না করলে সংকলন করার সময় সতর্ক বার্তা (warning message) আসতে পারে, আর ক্রমলেখ চালানোর সময় উল্টোপাল্টা ফলও আসতে পারে।

সবশেষে খেয়াল করো ফলন (output) দেওয়া হয়েছে যেখানে সেখানে উদ্ধৃতি চিহ্ন "" এর ভিতরে যা আছে তা কিন্তু মালা (string)। কাজেই ওইটা কিন্তু ওইভাবেই ফলনে এসেছে এমনকি khetrofol কথাটাও হুবহু এসেছে যেটা কিনা চলকের নামের হুবহু একই রকম। কিন্তু "" উদ্ধৃতির বাইরে যখন khetrofol লেখা হয়েছে একই সারিতে পরের দিকে সেখানে কিন্তু আর khetrofol ফলনে আসে নি, এসেছে সেটাকে চলক ধরলে যে মান হওয়ার কথা সেই 15। কাজেই এটা মনে রাখবে যে চলকের নাম "" উদ্ধৃতির ভিতরে মালা আকারে থাকলে ওইটা আসলে চলকটাকে বুঝায়

না। নামটা যখন উদ্ধৃতির বাইরে থাকে তখন ওইটা একটা নাম হয়, এইক্ষেত্রে একটা চলকের নাম হয় আর ওইটার মান নিয়ে কাজ হয়। একই অবস্থা porishima এর ক্ষেত্রেও। উদ্ধৃতি চিহ্নের ভিতরে থাকা porishima কথাটি হুবহু ফলনে এসেছে কিন্তু উদ্ধৃতির বাইরে থাকা porishima কথাটির বদলে ওটিকে চলক ধরলে যে মান পাওয়া যাবে তা ফলনে এসেছে।

২.২ ধ্রুবকের ব্যবহার (Using Constants)

একটি বৃত্তের ব্যাসার্ধ দেয়া আছে ৫ সেমি, বৃত্তটির ক্ষেত্রফল নির্ণয়ের জন্য সিপিপিতে একটি ক্রম-লেখ (program) রচনা করো। তোমার ক্রমলেখতে তুমি ব্যাসার্ধের জন্য একটি পূর্ণক (integer) ব্যবহার করবে। আর ক্ষেত্রফলের জন্য প্রথমে পূর্ণক ব্যবহার করে দেখবে কী হয়, তারপর ভগ্নক (fraction) অর্থাৎ সচলবিন্দু সংখ্যা (floating-point number) বা float ব্যবহার করবে। তুমি তো জানো বৃত্তের ক্ষেত্রফল হিসাব করার জন্য আমাদের পাইয়ের মান লাগবে। আমরা ওইটা সরাসরি সংখ্যায় না দিয়ে একটা ধ্রুবক (constant) হিসাবে ব্যবহার করবো, কারণ পাইয়ের মান তো কখনো বদলাবে না, সব সময় ধ্রুবক থাকবে। পাইয়ের মান যেহেতু ভগ্নক আমাদের ধ্রুবকটি তাই হবে float ধ্রুবক। চলো আমরা এবার তাহলে ক্রমলেখটি দেখি।

ফিরিস্তি ২.২: ক্রমলেখতে ধ্রুবকের ব্যবহার (Constants in Programs)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int bashardho = 5;  // এটি বৃত্তের ব্যাসার্ধের জন্য চলক

  float const pai = 3.1415; // পাইয়ের মানের জন্য ধ্রুবক

  // নীচে আমরা বৃত্তের ক্ষেত্রফলের সূত্র লিখছি
  int khetrofol = pai * bashardho * bashardho;

  // এবার আমরা পর্দায় ফলন দেখাবো।
  cout << "britter khetrofol " << khetrofol
       << " borgo cm" << endl;

  return EXIT_SUCCESS;
}
```

ফলন (output)

```
britter khetrofol 78 borgo cm
```

তো হয়ে গেল আমাদের বৃত্তের ক্ষেত্রফল নির্ণয়ের ক্রমলেখ। এই রকম চলক (variable) আর ধ্রুবক (constant) ব্যবহার না করেই তুমি কিন্তু ক্রমলেখ লিখতে পারতে, তাই না! আমরা কিন্তু সেটা পাঠ ২.১ এই আলোচনা করেছি। সেক্ষেত্রে main() বিপাতকের { } বন্ধনী দুটোর মধ্যে return EXIT_SUCCESS; এর আগে মাত্র এক সারিতে cout << "britter khetrofol " << 3.1415 * 5 * 5 << " borgo cm" << endl; লিখলেই আমাদের কাজ হয়ে যেতো। কিন্তু আমরা সেটা না করে কেন চলক ব্যবহার করছি সেটাও ওই একই পাঠে আলোচনা করেছি।

এবার আসি আমরা যে ক্রমলেখটি লিখলাম সেটার বিস্তারিত আলোচনায়। আমরা #include, using namespace, int main(), return ইত্যাদি সম্পর্কে ইত্যমধ্যে জেনেছি আগের পাঠ-গুলো থেকে, কাজেই আমাদের আলোচনা সীমাবদ্ধ থাকবে main() বিপাতকে আর যা যা লিখেছি সেই বিষয়গুলোতে। তো চলো আমরা এবার সারির পরে সারি ধরে আলোচনা করি।

উপরে যেমন বলা হয়েছে, সেই অনুযায়ী আমরা প্রথমে ব্যাসার্ধের জন্য একটা চলক নিয়েছি bashardho নামে যেটি হবে int ধরনের অর্থাৎ পূর্ণক বা পূর্ণ সংখ্যা। বৃত্তের ব্যাসার্ধ যদি তোমার ভগ্নক হয়, তুমি চাইলে int ব্যবহার না করে float ব্যবহার করতে পারো। আগের পাঠের সাথে এই পাঠে একটা বিষয় খেয়াল করো, আমরা কিন্তু ব্যাসার্ধ bashardho চলকের মান আলাদা সারিতে না দিয়ে যেখানে চলক ঘোষনা (variable declare) করেছি সেখানেই = চিহ্ন দিয়ে মান আরোপ করেছি অর্থাৎ bashardho এর মান সরাসরি 5 হয়ে গেছে। এটাকে বলা হয় চলকের আদি মান আরোপণ (initial value assignment)। এটা করার দুটো সুবিধা: একটা হলো আমাদের দুইটা আলাদা সারিতে দুইবার লিখতে হলো না, আরেকটা হলো চলকে উল্টাপাল্টা মান থাকার কারণে ক্রমলেখয়ে ভুল হিসাব করার সম্ভাবনা কমে গেল। জেনে রাখো চলক ঘোষনার সাথে সাথে কোন মান না দিয়ে না দিলেও ওখানে উল্টা পাল্টা একটা মান থাকে, কী মান থাকবে আমরা কিন্তু কোন ভাবেই আগে থেকে সেটা জানিনা, পুরাই উল্টাপাল্টা একটা মান হতে পারে সেটা। আর ভুলক্রমে ওই চলকে যদি পরে আর মান আরোপ (assign) করা না হয়, অথবা যদি আরোপ করার আগেই অন্য কোন হিসাবে চলকটি ব্যবহার করা হয়, তাহলে সঙ্গত কারণেই উল্টাপাল্টা মানটি কাজে লাগিয়ে একটা উল্টাপাল্টা ফলাফল আসবে, যেটা আমরা কখনোই চাই না।

ব্যাসার্ধের জন্য চলক নেয়ার পরে আমরা পাইয়ের মান রাখার জন্য একটি float const ধরনের ধ্রুবক নেবো যার নাম pai। পাইয়ের মান যেহেতু ভগ্ন সংখ্যা আমাদের তাই float নিতে হবে, আর পাইয়ের মান যেহেতু সব সময় ধ্রুবক তাই আমরা float এর পরে const লিখে দিতে চাই। তুমি যদি const না লিখো তাহলে কিন্তু এটা একটা চলকের মতো কাজ করবে।

```
int cholok = 15;  // একটা চলক ঘোষনা করে যার মান দিলাম 15
int const dhrubok = 20; // একটা ধ্রুবক ঘোষনা করলাম মান 20

// এখন পর্যন্ত চলক cholok এর মান 15, নীচে নতুন মান দেবো 23
// আবার মান আরোপ না করা পর্যন্ত cholok এর মান থাকবে 23

cholok = 23;     // এটা করা যাবে

// এখন পর্যন্ত ধ্রুবক dhrubok এর মান 20, নীচে নতুন মান দেবো 25
// কিন্তু ক্রমলেখ সংকলন (compile) করার সময় আমরা ত্রুটিবার্তা পাবো।
// ideone.com দিয়ে সংকলন করলে ত্রুটিবার্তাটি নিম্নরূপ হতে পারে
// error: assignment of read-only variable 'dhrubok'

dhrubok = 25;    // এটা করা যাবে না, ত্রুটি বার্তা আসবে
```

উপরের ক্রমলেখ লক্ষ্য করো। চলক আর ধ্রুবকের মধ্যে তফাৎ হলো চলকের (variable) মান ঘোষনার সময় একবার আরোপ করা যায়, আর তারপরেও যতবার ইচ্ছা ততবার নতুন নতুন মান আরোপ (assign) করা যায়। কিন্তু ধ্রুবকে (constant) একটা মান কেবল ঘোষনা করার সময় বলে দেওয়া যায়, ক্রমলেখতে পরে আর কোথাও ওই ধ্রুবকের মান বদলে নতুন মান আরোপ (assign) করা যায় না। যদি করো তাহলে সংকলক (compiler) ত্রুটি বার্তা (error message) দেখাবে। তো আমরা যেহেতু জানি যে পাইয়ের মান সবসময় ধ্রুবক, এটার মান আমাদের কখনো বদল হবে না, আমরা তাই এটাকে চলক হিসাবে ঘোষনা না করে ধ্রুবক হিসাবে ঘোষনা করবো।

আশা করা যায় চলক আর ধ্রুবকের পার্থক্য পরিস্কার হয়েছে। এবার দেখো আমাদের বৃত্তের ক্ষেত্রফলের ক্রমলেখতে আমরা ক্ষেত্রফলের জন্য khetrofol নামে একটা চলক নিয়েছি, যার প্রকরণ হল int বা পূর্ণক. যদিও আমরা জানি পাইয়ের মান ভগ্নক হওয়ার কারণে আমাদের ফলাফল আসলে একটি ভগ্নক হবে। এইটা আমরা মুলত পরীক্ষামুলক করছি। তো int নেয়ার কারণে আমরা আমাদের ক্রমলেখয়ের ফলন দেখতে পাবো 78 আসলে হওয়ার কথা 78.5375। এইটা কেন হলো কারণ হলো প্রথমে 78.5375 ঠিক মতো ভিতরে ভিতরে হিসাব হয়ে যাবে, কিন্তু যখন khetrofol চলকের মধ্যে মানটা আরোপ (assign) হবে তখন যেহেতু পূর্ণ সংখ্যা বলে ভগ্নাংশটুকু ঢুকানো যাবে না, তাই ওইটা বাদ পরে যাবে (truncation)। আর মান যেটা আরোপ হবে সেটা হলো বাঁকী পূর্ণাংশটুকু বা 78। তো ভগ্নাংশ সহ সঠিক ক্ষেত্রফল পাওয়ার জন্য khetrofol এর সামনে int না লিখে float লিখে দাও তাহলে দেখবে ঠিক ঠিক 78.5375 ই ফলন হিসাবে চলে আসবে।

উপরের আলোচনায় আমরা তিনটা ব্যাপার শিখলাম: ১) আমরা চলক (variable) না ধ্রুবক (constant) ব্যবহার করবো সেটা; তারপর ২) ঘোষনা করার সাথে সাথে একটা আদি মান দিয়ে দেয়া যাকে বলা হয় আদি মান আরোপণ (initial assignment), আর ৩) কোন চলক বা ধ্রুবকের প্রকরণ কেমন হবে, int না float হবে, পূর্ণক না ভগ্নক হবে সেটা আগে থেকে ধারনা করতে পারতে হবে, আর সেই অনুযায়ী চলক বা ধ্রুবকের প্রকার বলে দিতে হবে, না হলে সঠিক ফলাফল নাও পাওয়া যেতে পারে, যেমন 78.5375 এর বদলে 78 পাওয়া যেতে পারে।

২.৩ চলক ঘোষনা (Variable Declarations)

এই পাঠে সিপিপিতে একাধিক চলক (variable) আমরা কী ভাবে সহজে ঘোষনা (declaration) করতে পারি তা আলোচনা করবো। আমরা আগে দেখেছি প্রতিটি চলক আলাদা আলাদা করে, এমনকি আলাদা আলাদা সারিতে ঘোষনা করতে। তো সুবিধার জন্য আমরা চাইলে একাধিক চলক এক সারিতেই একটা বিবৃতিতেই ঘোষনা করতে পারি, যদি তাদের সকলের উপাত্ত প্রকরণ (data type) একই হয়, যেমন ওই চলকগুলোর সবই যদি int ধরনের হয় অথবা float ধরনের হয়। উদাহরণ দিয়ে ব্যাপারগুলো পরিস্কার করা যাক। ধরো আমরা doirgho, prostho, porishima নামে তিনটি চলক নিলাম, তিনটা চলকের প্রকরণই int অর্থাৎ পূর্ণক বা পূর্ণসংখ্যা।

```
int doirgho;  // দৈর্ঘ্যের জন্য চলক যা int ধরনের অর্থাৎ পূর্ণক
int prostho;  // প্রস্থের জন্য চলক যা int ধরনের অর্থাৎ পূর্ণক
int porishima;  // পরিসীমার জন্য চলক যা int ধরনের অর্থাৎ পূর্ণক
```

উপরের তিনটি চলকই যেহেতু int ধরনের, কাজেই আমরা ওই তিনটি চলককে চাইলে একটি বিবৃতিতেই (statement) ঘোষনা করতে পারি। সেক্ষেত্রে আমাদের int একবার লিখতে হবে, আর চলকগুলোর নাম একটার পর একটা বির্তি , (comma) দিয়ে লিখতে হবে।

```
int doirgho, prostho, porishima;// সবগুলোই int ধরনের
```

এবার আর একটি উদাহরণ দেখি, যেখানে বৃত্তের ব্যাসার্ধ আর ক্ষেত্রফল বের করতে হবে। তো ব্যাসার্ধ যদি int ধরনের বা পূর্ণক হয় আর ক্ষেত্রফল তো float ধরনের বা ভগ্নক হবেই। কাজেই আমরা এ দুটোকে একটা বিবৃতি (statement) দিয়ে ঘোষনা করতে পারবো না।

```
int bashardho;     // ব্যাসার্ধের জন্য চলক int ধরনের।
float khetrofol;   // ক্ষেত্রফলের জন্য চলক float ধরনের
```

কিন্তু যদি porishima এর মতো bashardho টাও float বা ভগ্নক ধরনের হতো তাহলে আমরা এক বিবৃতি দিয়েই দুটোকে এক সাথে ঘোষনা করতে পারতাম।

```
float bashardho, khetrofol; // ব্যাসার্ধ ও ক্ষেত্রফলের চলক
```

তাহলে একটা ক্রমলেখতেই (program) যদি আমরা আয়তের পরিসীমা আর বৃত্তের ক্ষেত্রফল বের করতে চাই, আমরা দরকারী সবগুলো চলক নীচের মতো করে ঘোষনা করতে পারি, যেখানে int চলকগুলো একটা বিবৃতিতে (statement) থাকবে আর float চলকগুলো আলাদা আরেকটা বিবৃতিতে থাকবে। মনে রেখো আমরা কিন্তু এই পাঁচটি চলকের প্রত্যেককে আলাদা আলাদা বিবৃতিতে লিখতেই পারতাম। এখানে আমরা সেটা না করা নিয়েই আলোচনা করছি।

```
int doirgho, prostho, porishima;  // দৈর্ঘ্য, প্রস্থ, পরিসীমা
float bashardho, khetrofol; // ব্যাসার্ধ ও ক্ষেত্রফল
```

২.৪ আদিমান আরোপণ (Initial Assignment)

আগের পাঠে আমরা একাধিক চলক (variable) ঘোষনা (declaration) নিয়ে আলোচনা করেছি। এখন আমরা এদের আদি মান (initial value) আরোপ (assign) করার দিকে নজর দেই। আদিমান হল প্রথমবারের মতো যে মান দিয়ে দেওয়া হয় সেই মানটি। ঘোষনা দেয়ার পরে চলকে আলাদা করে আদি মান আরোপ করতে চাইলে আমরা নীচের মতো করে করবো।

```
doirgho = 6;
prostho = 3;
bashardho = 5;
```

অথবা চাইলে এক বিবৃতিতে এক সাথেও করা সম্ভব, বির্তি , (comma) দিয়ে।

```
doirgho = 6, prostho = 3, bashardho = 5;
```

আমরা কিন্তু চাইলে আদিমানগুলো নীচের মতো ঘোষনার সাথে সাথেই দিতে পারতাম।

```
int doirgho = 6, prostho = 3, porishima;
float bashardho = 5, khetrofol;
```

ঘোষনার সাথে সাথে চলকের আদিমান দিলে ক্রমলেখয়ের (program) দক্ষতা অল্প একটু বাড়তে পারে। কারণ ঘোষনার সাথে সাথে আদিমান না দিলেও একটা উল্টাপাল্টা মান তো ভিতরে ভিতরে দেয়াই হয়, পরে যখন আমরা আবার মান দেই, তখন আরেকবার দেওয়া হলো, প্রথমবারেই দেওয়া হলো না। আর ঘোষনার সাথে সাথে আদিমান দিলে, একদম প্রথমবারেই মানটি চলকে দেওয়া হয়ে গেলো। ধ্রুবকের ক্ষেত্রে কিন্তু আদি ও একমাত্র মান ঘোষনার সাথে সাথেই দিতে হবে, পরে দেয়ার কোন সুযোগ নাই, সংকলক (compiler) ত্রুটি বার্তা দেখাবে।

২.৪. আদিমান আরোপণ (Initial Assignment)

কোন চলক ঘোষনার সাথে সাথে তাতে কোন আদিমান না দিলেও যে উল্টাপাল্টা মান থাকে সেটা কত তা যদি জানতো চাও তবে পরীক্ষা করে দেখতে পারো। ধরো তোমার চলক doirgho। এখন ঘোষনার পরেই cout << "doirgho holo " << doirgho << endl; লিখে ক্রম-লেখ সংকলন (compile) করে চালিয়ে (run) দেখতে পারো। কিন্তু মনে রাখবে প্রতিবার চালালে যে একই মান আসবে তার কোন নিশ্চয়তা নাই, যদি আসে সেটা নেহায়েত কাকতাল।

আমরা আগেই জানি বৃত্তের ক্ষেত্রফল নির্ণয়ের জন্য আমাদের পাইয়ের মান দরকার হবে, যেটি একটি ধ্রুবক (constant) আর পাইয়ের মান আসলেই ভগ্নক বা float। কিন্তু float হওয়া সত্ত্বেও আমরা কিন্তু পাইয়ের জন্য pai নামক চলকটিকে bashardho আর khetrofol এর সাথে একই বিবৃতিতে ঘোষনা করতে পারবো না। কারণ bashardho ও khetrofol হল চলক (variable) যাদের মান পরে যতবার ইচ্ছা বদলানো যাবে আর pai হল ধ্রুবক (constant) যার মান একবার দেওয়ার পরে আর বদলানো যাবে না। পাইয়ের মান তাই আলাদা করে ঘোষনা করতে হবে।

```
int doirgho = 6, prostho = 3, porishima;
float bashardho = 5, khetrofol;
float const pai = 3.1415; // পাইয়ের মানের জন্য ধ্রুবক
```

আমাদের যদি একাধিক float constant থাকে সেগুলোকে আবার এক বিবৃতিতেই ঘোষনা করতে পারবো, যেমন পাই আর জি এর মান ঘোষনা করছি নীচে। তোমরা জানো জি হল মাধ্যাকর্ষনের ত্বরণের মান, যা নির্দিষ্ট স্থানে মোটামুটি একটা ধ্রুবক।

```
float const pai = 3.1415, g = 9.81;
```

পরিসীমা আর ক্ষেত্রফলের জন্য আমাদের সূত্র লিখতে হবে, সেগুলোকে বির্তি , (comma) দিয়েই এক বিবৃতিতে লেখা সম্ভব, যেমন নীচে লিখলাম।

```
int doirgho = 6, prostho = 3
int porishima = doirgho * prostho;
float bashardho = 5;
float khetrofol = pai * bashardho * bashardho;
float const pai = 3.1415;
```

উপরে যা লিখলাম তাতে কিন্তু একটা ত্রুটি আছে, সংকলন (compile) করতে গেলেই ত্রুটি ধরা পড়বে। ত্রুটিটি হল আমরা pai ঘোষনা করেছি পঞ্চম বিবৃতিতে, কিন্তু pai ব্যবহার করেছি চতুর্থ বিবৃতিতে khetrofol এর সুত্র লিখতে গিয়েই। কোন চলক ঘোষনা করার আগে সেটা ব্যবহার করা যাবে না, সংকলক যখন চলে (run) তখন সে একে একে বিবৃতিগুলো উপর থেকে নীচে আর বামে থেকে ডানে পড়তে থাকে। তো কোন চলক বা ধ্রুবক ঘোষনার আগেই যদি তাদের ব্যবহারটা পড়ে ফেলে যেমন pai, তাহলে সে বুঝতে পারবে না pai টা কী জিনিস, এইটা কি চলক নাকি ধ্রুবক, এটা কি int ধরনের নাকি float ধরনের। আমাদের তাই ঘোষনা অবশ্যই আগে করতে হবে, ব্যবহার করতে হবে পরে। তো চলো নীচে আমরা ঠিক করে পাইয়ের ঘোষণা আগে লিখি।

```
int doirgho = 6, prostho = 3;
int porishima = doirgho * prostho;
float const pai = 3.1415; // ঘোষনা আগে করা হলো
float bashardho = 5;
float khetrofol = pai * bashardho * bashardho;
```

লক্ষ্য করো doirgho, prostho, bashardho এর জন্য কিন্তু উপরের ওই ত্রুটি ঘটে নি, কারন সুত্রে ব্যবহারের আগেই তো ওগুলো ঘোষনা হয়েছে, যদিও একই সারিতে কিন্তু বামের বিষয়গুলো যেহেতু ডানেরগুলোর থেকে আগে, তাই ঘোষনা আগেই হয়েছে। আমরা অবশ্য উপরের মতো করে সুত্রও একই বিবৃতিতে না দিতে বলবো। তাতে পড়ারও সুবিধা হয়, আবার আগে লিখবো না পরে লিখবো সেই সমস্যাও দুর হয়। তাহলে পুরো ব্যাপারটি দাঁড়াচ্ছে নীচের মতো:

```
int doirgho = 6, prostho = 3, porishima;
float const pai = 3.1415;
float bashardho = 5, khetrofol;

porishima = doirgho * prostho;
khetrofol = pai * bashardho * bashardho;
```

২.৫ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. ক্রমলেখতে (program) চলক ও ধ্রুবক কেন ব্যবহার করা হয়? কারণগুলো উল্লেখ করো।

২. ক্রমলেখতে (program) চলক ঘোষনা বলতে কী বুঝ, যথাযথ উদাহরণ দিয়ে দেখাও।

৩. চলকে (variable) মান আরোপণ (value assign) বলতে কী বুঝ? ব্যাখ্যা করো।

৪. কখন তুমি চলক (variable) ব্যবহার না করে ধ্রুবক (constant) ব্যবহার করবে?

৫. সিপিপিতে কী ভাবে চলক ও ধ্রুবক ঘোষনা করতে হয়। যথাযথ উদাহরণ দিয়ে দেখাও।

৬. সিপিপিতে কী ভাবে পূর্ণক ও ভগ্নক ধরনের চলক ঘোষনা করতে হয় উদাহরণ দিয়ে দেখাও।

৭. সিপিপিতে এক সারিতে কখন একাধিক চলক ঘোষনা করা যায়? উদাহরণ দিয়ে দেখাও।

৮. চলকে (variable) আদিমান আরোপণ (initial value assignment) কী?

৯. চলকে (variable) আদিমান আরোপণ না করলে সম্ভাব্য কী ফলাফল ঘটতে পারে?

১০. ধ্রুবকে (constant) কেন আদিমান আরোপ করতে হয়, কিন্তু পরে আরোপ করা যায় না?

১১. ফলাফল ভগ্নক (float) কিন্তু int ধরনের পূর্ণক চলকে আরোপ করলে কী ঘটে?

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. এমন একটি ক্রমলেখ রচনা করো যেটি দুটি পূর্ণক int ধরনের, আর একটি ভগ্নক float ধরনের চলক ঘোষনা করে। ক্রমলেখটি তারপর চলক তিনটির মান যথাক্রমে 10, 15, 12.6 আরোপণ করে। পরিশেষে ক্রমলেখটি চলকগুলোর মান পর্দায় দেখায়।

২.৫. অনুশীলনী সমস্যা (Exercise Problems)

২. ধরো দুটো পূর্ণ সংখ্যা 89 আর 56। এই দুটিকে তুমি দুটো চলকে নিবে, আর তারপর দুইটি চলকে তাদের যোগফল, বিয়োগফল নির্ণয় করবে। সবশেষে সবগুলো চলকের মান ফলনে দেখাবে। সব মিলিয়ে একটি ক্রমলেখ রচনা করো।

৩. যদি তাপমাত্রা সেলসিয়াসে c ডিগ্রী হয় আর ফারেনহাইটে হয় f ডিগ্রী, তাহলে আমরা সুত্র লিখতে পারি f = 9c/5 + 32। ধরো তাপমাত্রা 30 ডিগ্রী সেলসিয়াস, তাহলে ফারেনহাইটে এটি কত হবে? তোমার ক্রমলেখতে তুমি ভগ্নক float ধরনের চলক ব্যবহার করবে।

৪. যদি তাপমাত্রা ফারেনহাইটে হয় f ডিগ্রী আর সেলসিয়াসে হয় c ডিগ্রী, তাহলে আমরা সুত্র লিখতে পারি c = 5(f - 32)/9। ধরো তাপমাত্রা 76 ডিগ্রী ফারেনহাইট, তাহলে সেলসিয়াসে এটি কত হবে? তোমার ক্রমলেখতে তুমি ভগ্নক float ধরনের চলক ব্যবহার করবে।

৫. ধরো একটা কাজ করতে তোমার 7 ঘন্টা 15 মিনিট 39 সেকেন্ড লেগেছে। এই সময়কে সে-কেন্ডে রূপান্তর করো। তোমার ক্রমলেখতে তুমি 60 সেকেন্ডে এক মিনিট আর 60 মিনিটে এক ঘন্টা এই দুটি বিষয় বুঝানোর জন্য দুটো ধ্রুবক ব্যবহার করবে।

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. এমন একটি ক্রমলেখ রচনা করো যেটি দুটি পূর্ণক int ধরনের, আর একটি ভগ্নক float ধরনের চলক ঘোষনা করে। ক্রমলেখটি তারপর চলক তিনটির মান যথাক্রমে 10, 15, 12.6 আরোপণ করে। পরিশেষে ক্রমলেখটি চলকগুলোর মান পর্দায় দেখায়।

ফিরিস্তি ২.৩: চলক ঘোষনার ক্রমলেখ (Program Declaring Variables)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int purnok1, purnok2; //  পূর্ণক দুটি এক সাথে ঘোষনা
  float vognok;     //  ভগ্নকটি আলাদা ঘোষনা

  purnok1 = 10, purnok2 = 15; //  পূর্ণকে মান আরোপণ
  vognok = 12.6;          //  ভগ্নকে মান আরোপণ

  cout << "purnok duti ";    // এখানে endl দেই নাই
  cout << purnok1 << " " << purnok2 << endl;
  cout << "vognok holo " << vognok << endl;

  return EXIT_SUCCESS;  //  সফল সমাপ্তি
}
```

ফলন (output)

```
purnok duti 10 15
vognok holo 12.6
```

২. ধরো দুটো পূর্ণ সংখ্যা 89 আর 56। এই দুটিকে তুমি দুটো চলকে নিবে, আর তারপর দুইটি চলকে তাদের যোগফল, বিয়োগফল নির্ণয় করবে। সবশেষে সবগুলো চলকের মান ফলনে দেখাবে। সব মিলিয়ে একটি ক্রমলেখ রচনা করো।

আমরা এই ক্রমলেখতে কেবল দরকারী অংশটুকু দেখাচ্ছি। ধরে নিচ্ছি যে তুমি দরকারী শির নথি (header) অন্তর্ভুক্ত করা, নামাধার std ব্যবহার, main বিপাতক লেখা ও মান ফেরত দেয়া ইত্যমধ্যে ভালো করে শিখে ফেলেছো। তো তুমি যদি সত্যি নীচের লেখা ক্রমলেখাংশ সংকলন করে চালাতে চাও, তোমাকে কিন্তু আগে include, namespace, main, return ওইগুলো লিখে নিতে হবে, তারপর main বিপাতকের ভিতরে return এর আগে তুমি আমাদের নীচের অংশগুলো লিখে নিবে। তারপর সংকলন করে ক্রমলেখ চালাবে।

ফিরিস্তি ২.৪: পাটিগণিতের অণুক্রিয়ার ক্রমলেখ (Arithmetic Program)

```
int prothom = 89, ditiyo = 56;

int jogfol = prothom + ditiyo;
int biyogfol = prothom - ditiyo;

cout << "prothom holo " << prothom;
cout << " ditiyo holo " << ditiyo;
cout << endl;

cout << "jogfol holo " << jogfol;
cout << " biyogfol holo " << biyogfol;
cout << endl;
```

৩. যদি তাপমাত্রা সেলসিয়াসে c ডিগ্রী হয় আর ফারেনহাইটে হয় f ডিগ্রী, তাহলে আমরা সুত্র লিখতে পারি f = 9c/5 + 32। ধরো তাপমাত্রা 30 ডিগ্রী সেলসিয়াস, তাহলে ফারেনহাইটে এটি কত হবে? তোমার ক্রমলেখতে তুমি ভগ্নক float ধরনের চলক ব্যবহার করবে।

ধরে নিচ্ছি প্রথম সমস্যার সমাধান দেখে তুমি ক্রমলেখয়ের কাঠামো দাঁড় করতে পারবে।

ফিরিস্তি ২.৫: সেলসিয়াস থেকে ফারেনহাইটে রূপান্তর (Celcius to Fahrenheit)

```
float c = 30, f = 9 * c / 5 + 32;
```

৪. যদি তাপমাত্রা ফারেনহাইটে হয় f ডিগ্রী আর সেলসিয়াসে হয় c ডিগ্রী, তাহলে আমরা সুত্র লিখতে পারি c = 5(f - 32)/9। ধরো তাপমাত্রা 76 ডিগ্রী ফারেনহাইট, তাহলে সেলসিয়াসে এটি কত হবে? তোমার ক্রমলেখতে তুমি ভগ্নক float ধরনের চলক ব্যবহার করবে।

ধরে নিচ্ছি প্রথম সমস্যার সমাধান দেখে তুমি ক্রমলেখয়ের কাঠামো দাঁড় করতে পারবে।

ফিরিস্তি ২.৬: ফারেনহাইট থেকে সেলসিয়াসে রূপান্তর (Fahrenheit to Celcius)

```
float f = 76, c = 5*(f - 32) / 9;
```

৫. ধরো একটা কাজ করতে তোমার 7 ঘন্টা 15 মিনিট 39 সেকেন্ড লেগেছে। এই সময়কে সেকেন্ডে রূপান্তর করো। তোমার ক্রমলেখতে তুমি 60 সেকেন্ডে এক মিনিট আর 60 মিনিটে এক ঘন্টা এই দুটি বিষয় বুঝানোর জন্য দুটো ধ্রুবক ব্যবহার করবে।

ধরে নিচ্ছি প্রথম সমস্যার সমাধান দেখে তুমি ক্রমলেখয়ের কাঠামো দাঁড় করতে পারবে।

ফিরিস্তি ২.৭: সময়কে সেকেন্ডে রূপান্তর (Convert Time to Seconds)

```
int ghonta = 7, minit = 15, sekend = 39;
int const ghontaiMinit = 60, miniteSekend = 60;

int motMin = ghonta * ghontaiMinit + minit;
int motSek = motMin * miniteSekend + sekend;
```

২.৬ গণনা পরিভাষা (Computing Terminologies)

- আদি মান (initial value)
- আরোপণ (assign)
- ঘোষনা (declaration)
- চলক (variable)
- চলক ঘোষণা (variable declaration)
- ধ্রুবক (constant)
- ভগ্নক (fraction)
- মান (value)
- মান আরোপণ (value assign)
- সচলবিন্দু (floating-point)

অধ্যায় ৩

শনাক্তকের নামকরণ (Naming Identifiers)

নামে কী আসে যায় কর্মে পরিচয়। আপনার কাজই নির্ধারণ করে দেবে আপনার পরিচয়। আপনার নাম পরিচয় হবে আপনার কাজের কারণেই। ক্রমলেখ (program) লিখতে গিয়ে আমরা তাই চলক (variable), ধ্রুবক (constant), বিপাতক (function) সহ যে কোন কিছুর নাম দেই তাদের কী কাজে লাগানো হবে সেটা মাথায় রেখে।

৩.১ গঠনসিদ্ধ নাম (Legitimate Names)

সিপিপিতে চলক ও ধ্রুবকের ব্যবহার তুমি ইত্যমধ্যে শিখে ফেলেছো। আর চলকের নাম কী রকম দিতে হবে সেটাও আগে একটু জেনেছো। এখন আমরা বিস্তারিত ভাবে শিখব সিপিপিতে কী ভাবে চলক বা ধ্রুবকের নাম দিতে হয়, বিশেষ করে নামের গঠনরীতি (syntax) কেমন অর্থাৎ নামে কী রকম অক্ষর থাকতে পারবে অথবা পারবে না। আমরা আপাতত কেবল main বিপাতক (function) নিয়ে কাজ করছি। কিন্তু ভবিষ্যতে আমরা যখন নিজেদের জন্য নানান বিপাতক তৈরী করবো, তখন বিপাতকের নামকরণের জন্যেও চলক বা ধ্রুবকের নাম তৈরীর নিয়মগুলোই কাজে লাগবে। চলক বা ধ্রুবক বা বিপাতক যাইহোক নাম কে বলা হয় শনাক্তক (identifier)।

সিপিপিতে কোন শনাক্তকের (identifier) নামে কেবল ১) ইংরেজী বর্ণমালার বড় হাতের অক্ষর A-Z, ২) ইংরেজী বর্ণমালার ছোট হাতের অক্ষর a-z, ৩) ইংরেজী অংক 0-9 আর ৪) নিম্নদাগ (underscore) _ থাকতে পারবে। তবে শনাক্তকের নামের প্রথম অক্ষর আবার অংক 0-9 হতে পারবে না, প্রথম অক্ষর ছাড়া অন্য যে কোন অক্ষর হিসাবে অংকগুলো ব্যবহার করা যাবে। সুতরাং বোঝাই যাচ্ছে প্রথম অক্ষর যে কোন বর্ণ A-Z বা a-z অথবা নিম্নদাগ (underscore) _ হতে পারবে। আর তারপরের যে কোন অক্ষর বর্ণ বা অংক বা নিম্নদাগ হতে পারবে। সিপিপিতে শনাক্তকের নামের দৈর্ঘ্য নিয়ে কোন বিধিনিষেধ নেই তবে ক্রমলেখ (program) সংকলনে (compile) কী সংকলক ব্যবহার করা হচ্ছে তার ওপর এটা নির্ভর করতে পারে। ideone.org দিয়ে সংকলন করলে কোন বিধি নিষেধ নেই, মাইক্রোসফট c++ দিয়ে সংকলন করলে ২০৪৮ অক্ষর পর্যন্ত হতে পারে। যাইহোক আমরা এখানে গঠনরীতি অনুযায়ী বৈধ ও অবৈধ কিছু নাম দেখবো।

অবৈধনাম	কারণ
12	নামের সবগুলোর অক্ষর অংক হতে পারবে না
12cholok	নামের প্রথম অক্ষর অংক হতে পারবে না
amar cholok	নামের মাঝখানে কোন ফাঁকা (space) থাকতে পারবে না
ama;cho+k	বর্ণ, অংক, নিম্নদাগ ছাড়া অন্য কোন প্রতীক থাকতে পারবে না

ক্রমলেখতে (program) অবৈধনাম ব্যবহার করলে কী হয়? করে দেখো কী হয়! সংকলক (compiler) ত্রুটিবার্তা (error message) দিবে, আর তোমাকে নামটি ঠিক করতে হবে। তাহলে এখন থেকে তোমার ক্রমলেখতে নাম দেওয়ার সময় নামের এই গঠনরীতি গুলো মেনে চলবে।

বৈধনাম	কারণ
p	একটাই অক্ষর সেটি ছোট হাতের বর্ণ
P	একটাই অক্ষর সেট বড় হাতের বর্ণ
abc	তিনটা অক্ষর সব ছোট হাতের বর্ণ
ABC	তিনটা অক্ষর সব বড় হাতের বর্ণ
Abc	তিনটা অক্ষর ছোটহাতের বড়হাতের মিশানো
bAc	তিনটা অক্ষর ছোটহাতের বড়হাতের মিশানো
a1bc	তিনটা ছোটহাতের অক্ষর ও একটা অংক, অংকটি শুরুতে নয়
a1Bc	তিনটা ছোটবড় হাতের অক্ষর ও একটা অংক যেটি শুরুতে নয়
a_bc	তিনটা ছোটহাতের অক্ষর ও একটি নিম্নদাগ
_abc	তিনটা ছোট হাতের অক্ষর ও তিনটি নিম্নদাগ
_A_b_c	তিনটা ছোটবড় হাতের অক্ষর ও তিনটি নিম্নদাগ
amar_cholok	ছোটহাতের অক্ষর ও নিম্নদাগ, নামটি অধিক বোধগম্য
_amar_Cholok	ছোটবড় হাতের অক্ষর ও নিম্নদাগ, অধিক বোধগম্য
_amarCholok123	ছোটবড় হাতের অক্ষর, নিম্নদাগ, ও অংক যেটি শুরুতে নয়
amar125cholok	ছোটহাতের অক্ষর ও অংক, অংকটি শুরুতে নয়।

৩.২ অর্থবোধক নাম (Meaningful Names)

সিপিপিতে শনাক্তকের (identifier) নাম কেমন হতে পারে আর কেমন হতে পারে না, আমরা তা আগের পাঠে দেখেছি। এই পাঠে আমরা দেখবো নামের অর্থবোধকতা (semantic)। আমরা যখন কোন নাম দেবো, তখন নামটি অবশ্যই অর্থবহ হওয়া চাই। আমরা আগের একটি পাঠে অল্প একটু আলোচনা করেছি নামের অর্থবোধকতা নিয়ে। এখন আরো বিস্তারিত আলোচনা করছি নামগুলো কেমন হলে ভালো হয় সে সম্পর্কে। চলক (variable) বা ধ্রুবক (constant) বা বিপাতকের (function) নাম সবসময় তার কাজ ও ব্যবহারের দিকে খেয়াল রেখে অর্থবোধক হওয়া উচিৎ। অর্থবোধক না হলে ক্রমলেখ (program) বোঝা আমাদের জন্য কঠিন হয়ে যায়।

অনেকে অতিরিক্ত আগ্রহে যত্রতত্র নিজের নামে বা প্রিয় কারো নামে শনাক্তকের নামকরণ করে থাকে যেমন gonimia1, gonimia2, ইত্যাদি। তো এই চলক দুটোর একটা যদি ব্যাসার্ধের জন্য আরেকটা যদি ক্ষেত্রফলের জন্য ব্যবহার করা হয়, তাহলে চলকের নাম থেকে মোটেও বুঝা যাবে না কোন নামটি কী কাজে ব্যবহৃত হচ্ছে। ব্যাসার্ধের জন্য বরং radius বা bashardho অথবা নিদেনপক্ষে r বা b ব্যবহার করা যেতে পারে। এক অক্ষরের নাম দেয়া অনেকে পছন্দ করে, কারণ তাড়াতাড়ি লেখা যায়, কিন্তু একই আদ্যাক্ষর যুক্ত একাধিক চলক থাকলে তখন মুশকিল হয়ে যায়। সেক্ষেত্রে ওই অক্ষরের সাথে আরো অক্ষর লাগিয়ে অথবা সংখ্যা লাগিয়ে প্রতিটি নামকে আলাদা করতে হবে, যাতে অন্তত বুঝা যায় কোন চলকটি কী উদ্দেশ্যে ব্যবহার করা হয়েছে।

আমরা যদি দুটো বৃত্ত নিয়ে কাজ করি তাহলে তাদের ব্যাসার্ধের জন্য দুটি চলক হতেই পারে bashardho1 আর bashardho2 তাতে কোন সমস্যা নাই, বরং ভালো। ব্যাপারটা দীপু নাম্বার ২ চলচ্চিত্রে মতো, একজনের নাম দীপু নম্বর ১ আর একজন দীপু নম্বর ২। অথবা কেউ চাইলে নাম দিতে পারে bashardhoA আর bashardhoB। এভাবে একই ধরনের কাজে ব্যবহার হবে এরকম অনেকগুলো চলক লাগলে আমরা সংখ্যা লাগিয়ে বা বর্ণ লাগিয়ে আলাদা আলাদা নাম তৈরী করে নিবো। এর জন্য অবশ্য সাজন (array) নামে আলাদা একটা ধারণা আছে, যেটা আমরা পরে জানবো। সাজন ব্যবহার করে আমরা সংখ্যা লাগিয়ে যত ইচ্ছা ততগুলো একই ধরনের নাম পাই। অনেকে অতিরিক্ত অলস হয়ে অথবা যে কোন কারণে শনাক্তকের নাম করণ করতে থাকে a, b, c, p, q, r, i, j, k, x, y, z ইত্যাদি একের পর এক অক্ষর দিয়ে। এটা একটা খুবই বাজে অভ্যাস। এইরকম শনাক্তক মোটেও অর্থবোধক নয়। এগুলো থেকে বুঝার কোন উপায় নেই কোন চলকটি ঠিক কী কাজে ব্যবহার করা হচ্ছে। সবসময় এরকম নামকরণ থেকে দুরে থাকতে চেষ্টা করবে।

এখানে প্রশ্ন করতে পারো: নামকরণে কি সবসময় একটা মাত্র শব্দই ব্যবহার করবো? একের অধিক শব্দ ব্যবহার করবো না? উত্তর হচ্ছে অর্থবোধক করার জন্য তুমি দরকার মতো একাধিক শব্দ অবশ্যই ব্যবহার করবে, এইটা খুবই ভালো অভ্যাস। আর সেক্ষেত্রে যাতে প্রতিটি শব্দ খুব সহজে বোঝা যায় সে জন্য তোমার কিছু কৌশল অবলম্বন করতে হবে। একটা কৌশল হলো দুটি শব্দের মাঝে একটি নিম্নদাগ _ দেওয়া যেমন amar_cholok। আরেকটি কৌশল হল প্রতিটি শব্দের প্রথম অক্ষরটি বড়হাতের দেওয়া আর অন্যগুলো ছোট হাতের, যেমন AmarCholok তবে চাইলে একদম প্রথম শব্দের প্রথম অক্ষরটি ছোটহাতেরও রাখতে পারো যেমন amarCholok. নীচের সারণীতে আমরা কিছু অর্থবোধক নামের উদাহরণ দেখবো।

নাম	যথোপযুক্ততার কারণ
sum	যোগফলের জন্য sum চলকের ইংরেজী নাম
jogfol	যোগফলের জন্য jogfol চলকের বাংলা নাম
bijor_songkhar_jogfol	নিম্নদাগ _ দিয়ে অর্থবোধক শব্দ আলাদা হয়েছে
odd_number_sum	নিম্নদাগ _ দিয়ে অর্থবোধক শব্দ আলাদা হয়েছে
Bijor_Shongkhar_Jogfol	নিম্নদাগ _ দিয়ে আলাদা, বড়হাতের আদ্যাক্ষর
BijorShongkharJogfol	বড়হাতের প্রথম অক্ষর দিয়ে আলাদা আলাদা
bijorShongkharJogfol	এটি অনেক প্রচলিত ও অনেকেরই পছন্দের

৩.৩ লিপি সংবেদনশীলতা (Case Sensitivity)

সিপিপি ভাষা একটি লিপি সংবেদনশীল (case sensitive) ভাষা। এই কথার অর্থ কী? সিপিপিতে বড়হাতের ছোটহাতের অক্ষর কি ভিন্নভিন্ন ধরা হয়, নাকি ইংরেজীর মতো একই ধরা হয়?

```
barek is going home
BAREK IS GOING HOME
Barek Is Going Home
```

আগের কয়েকটি পাঠে চলক (variable) বাধ্রুবক (constant) বা বিপাতকের (function) নাম, এককথায় শনাক্তকের (identifier) নামকরণ নিয়ে আমরা আলোচনা করেছি। নামকরণের নিয়মগুলো আলোচনা করার সময় দেখেছি যে কোন শনাক্তকের নামকরণে আমরা চাইলে বড়হাতের বর্ণ A-Z, ছোটহাতের বর্ণ a-z, অংক 0-9, আর নিম্নদাগ _ ব্যবহার করতে পারবো। একই নামে বড়হাতের ছোটহাতের অক্ষর মিশিয়েও নামকরণ করতে পারবো। এমতাবস্থায় প্রশ্ন হচ্ছে কোন নাম ইচ্ছামতো একবার বড়হাতের অক্ষরে অথবা ছোট হাতের অক্ষরে অথবা আরেকবার কিছু অক্ষর

ছোটহাতের কিছু অক্ষর বড় হাতের এইভাবে লিখতে পারবো কিনা। বিশেষ করে আমরা জানি ইংরেজীতে আমি ছোট হাতেরই লিখি আর বড় হাতেরই লিখি শব্দটা আসলে একই থাকে, সিপিপিতেও কি তাই? আমরা বরং উদাহরণ দিয়ে ব্যাপারটা দেখি। ইংরেজীতে ছোট হাতের বড় হাতের অক্ষর আলাদা হলেও ওগুলো কেবলই সৌন্দর্যবর্ধন মুলক। উপরের তিনটে ইংরেজী বাক্য তাই একই।

এবার আমরা সিপিপি ভাষায় ছোট হাতের বড় হাতের অক্ষরের ব্যবহার দেখি। নীচের নামগুলোর প্রত্যকেটি সিপিপি ভাষায় আলাদা আলাদা নাম হিসাবে ধরা হবে।

```
amarcholok, amarCholok, AmarCholok, amar_cholok,
Amar_Cholok, amarChoLok, AmarChOlOk
```

সিপিপিতে উপরের একটা নাম দিয়ে যে চলক বা ধ্রুবক বা বিপাতককে বুঝানো হবে অন্য নাম দিয়ে ওইটাকে বুঝানো যাবে না, বরং অন্য একটা বুঝানো হয়ে যাবে। মোদ্দা কথা দুটো নামের একটা অক্ষরেও যদি এ দিক সেদিক থাকে তাহলে নামদুটো আসলে আলাদা। দুটোকে একই জিনিসের নাম হিসাবে ধরে নেয়া যাবে না। সুতরাং ক্রমলেখ (program) লেখার সময় খেয়াল রাখবে যাতে একটা চলককে বুঝাতে গিয়ে কেবল বড়হাতের ছোটহাতের বর্ণের ভিন্নতার কারণে আরেকটাকে বুঝিয়ে না ফেলো, তাতে সব ভজঘট লেগে যাবে। তোমার ক্রমলেখও উল্টাপাল্টা ফলাফল দিবে। আবার ধরো তোমার একটাই চলক যার নাম amarcholok, কিন্তু পরে তুমি লিখেছো amarCholok. এই অবস্থায় সংকলন (compile) করলে তোমাকে "amarCholok is not declared" এইরকম ত্রুটিবার্তা (error message) দিবে। তোমাকে তখন amarCholok এর বদলে amarcholok লিখে ঠিক করতে হবে। ক্রমলেখ তৈরীর সময় আমরা প্রায়শই এইরকম ভুল করে থাকি।

উপরের এই নিয়ম জানার পরে তুমি হয়তো মনে করবে এইটা তো ভালোই। আমার যদি দুইটা বৃত্তের ব্যাসার্ধের জন্য চলক লাগে একটার নাম দিবো bashardho আর একটার নাম দিবো Bashardho. হ্যাঁ, তুমি সেটা দিতেই পারো। সিপিপি যেহেতু দুইটাকে আলাদা আলাদা চলক হিসাবে ধরে নিবে, তাই এই দুটো হলো দুটো বৈধ আলাদা নাম। তবে অর্থবোধকতার দিক ভেবে তুমি হয়তো এরকম নাম করণ থেকে দূরে থাকার চেষ্টা করবে। একটা অক্ষর বড় বা ছোটহাতের কেবল এই অল্প একটুখানি ভিন্নতা দিয়ে আসলে তেমন বেশী অর্থবোধক পার্থক্য তৈরী করা যায় না, ফলে ক্রমলেখ (program) পড়া কঠিন হয়। আর একটা ব্যাপার: চলকের নামকরণে বড়হাতের ছোটহাতের অক্ষর মিশাতে তো পারোই যেমন AmarCholok, কিন্তু এমন ভাবে মিশিও না যে পড়াটা খুব কঠিন হয়ে যায়, যেমন AmArChOlOk, এই রকম নাম চট করে পড়া আসলে সম্ভব না, বরং এইরকম নাম যন্ত্রনাদায়ক। কাজেই সবমিলিয়ে সহজ ও সুন্দর নাম দিবে, কেমন!

৩.৪ সংরক্ষিত ও চাবি শব্দ (Reserved & Key Words)

সংরক্ষিত শব্দ (reserved word) বা চাবি শব্দ (key word) কী? আমি কি চলক (variable), ধ্রুবক (constant) বা বিপাতকের (function) এর শনাক্তক (identifier) হিসাবে সংরক্ষিত শব্দ বা চাবি শব্দ ব্যবহার করতে পারবো? সিপিপিতে সংরক্ষিত শব্দ বা চাবি শব্দ কোনগুলো?

সংরক্ষিত শব্দ বিষয়ে আলোচনার আগে আমরা একটা গল্প বলে নেই। এক বাড়িতে থাকে জামাই-বউ আর তাদের সাথে থাকে বড় কুটুম অর্থাৎ বউয়ের ভাই বা জামাইয়ের শ্যালক। তো সেই শ্যালকের নাম হল দুলাল। একদিন জামাই বেচারা তার বউয়ের কষ্ট লাঘব করার জন্য একজন কাজের ছেলে নিয়ে আসে। বউ জিজ্ঞেস করে "এই ছেলে তোমার নাম কী?" কাজের ছেলে বলে তার নাম দুলাল। বউ তখন জামাইকে বলে ছেলেটির নাম বদলাতে হবে। জামাই অবাক, অবাক কাজের ছেলেটিও। তার নাম দুলাল, ভালোই তো নামটি, সেটা বদলাতে হবে কেন। বউ জামাইকে বকতে থাকে "তুমি জানো না আমার ভাই অর্থাৎ তোমার শ্যালকের নাম দুলাল"। যে বাসায় শ্যালকের

নাম দুলাল, সেই বাসার কাজের ছেলের নাম দুলাল হয় কেমনে, শ্যালক হলো বড় কুটুম, তার কী এত বড় অসম্মান করা যায়! আর জামাইয়ের নাম হলো কাদের। তো বউ আরো এক কাঠি বাড়িয়ে বলতে থাকে ঠিক আছে কাজের ছেলের নাম বদলে কাদের রাখা হউক, দেখি জামাইয়ের কেমন লাগে। তারপর জামাইয়ের সামনেই কাজের ছেলেকে বলে "এই এখন থেকে তোর নাম দিলাম কাদের।" তারপর হেঁড়ে গলায় ডাকতে থাকে "কাদের, এই কাদের, এই দিকে আয়।" কেমন একটা বেড়াছেড়া অবস্থা। শেষ পর্যন্ত ঠিক হয় এক বাসায় দুইটা দুলাল তো হতে পারেনা, একজনের নাম বদলাতে হবে। আর বাসার বড় কুটুমের নাম তো আর বদলানো যাবে না কোন ভাবেই, ওটা সংরক্ষিত নাম, কাজেই বদলাতে হবে কাজের ছেলের নাম। সুতরাং কাজের ছেলের নাম দেয়া হয় দুলাল্যা। তাহলে শ্যালকের নাম দুলাল, আর কাজের ছেলের নাম দুলাল্যা।

সিপিপি ভাষায় গঠন কাঠামো ঠিক রাখার জন্য কিছু সুনির্দিষ্ট শব্দ আছে। আমরা ইত্যমধ্যে এরকম কিছু শব্দ ব্যবহার করেছি। যেমন return, int, float. এই শব্দগুলোর অর্থ সিপিপি ভাষাতে আগে থেকে সুনির্দিষ্ট, যেমন return মানে যখন বিপাতক (function) শেষ হয়, int আর float হল চলকের মান কেমন পূর্ণক বা পূর্ণ সংখ্যা না ভগ্নক বা ভগ্ন সংখ্যা এইরকম। এই তিনটি ছাড়াও আরো অনেকগুলো এই রকম শব্দ আছে। এই শব্দগুলো চাইলে আমরা নিজেরা আমাদের চলক বা ধ্রুবক বা বিপাতকের নাম হিসাবে ব্যবহার করতে পারবো না। এইগুলো হচ্ছে সংরক্ষিত শব্দ (reserved word)। এই শব্দগুলোকে অন্য কথায় চাবি শব্দও (key word) বলা হয়। তাহলে তোমার ক্রমলেখতে তুমি এইরূপ সংরক্ষিত শব্দ বা চাবি শব্দ শনাক্তকের (identifier) নাম হিসাবে ব্যবহার করবে না। কারণ ওগুলো বড়কুটুম দুলালের নামের মতো। যদি একান্তই দরকার হয় তাহলে দুলাল কে দুলাল্যা বানানোর মতো কিছু যোগ-বিয়োগ করে ভিন্ন শব্দ বানিয়ে ব্যবহার করবে। যেমন return না ব্যবহার করে returnValue ব্যবহার করলে, এইরকম। নীচে আমরা সিপিপির সংরক্ষিত শব্দগুলোর তালিকা দিচ্ছি।

- সংগঠিত পরিগণনায় (structured programming) ব্যবহৃত শব্দ:
  break, case, char, const, continue, default, do, double, else,
  enum, extern, float, for, goto, if, int, long, register, short,
  signed, sizeof, static, struct, switch, typedef, union, unsigned,
  return, void, volatile, while.

- বস্তুমুখী (object-oriented) পরিগণনায় ব্যবহৃত শব্দ:
  class, explicit, delete, friend, inline, mutable, namespace, new,
  opertor, private, protected, public, this, using, virtual

- ত্রুটি সামলানোর (error handling) জন্য শব্দ:
  catch, noexcept, throw, try

- যুক্তি ও বিটপ্রতি অণুক্রিয়ার (logical and bit-wise operators) শব্দ:
  bool, and, and_eq, bitand, bitor, compl, false, not, not_eq,
  or, or_eq, true, xor, xor_eq

- উপাত্ত প্রকরণ (data type) সংক্রান্ত শব্দ:
  auto, const_cast, decltype, nullptr, dynamic_cast,
  reinterpret_cast, static_cast, typeid

- ছাঁচ (template) সংক্রান্ত শব্দ:
  export, template, typename

- সংকলন সময়ে (compile-time) ব্যবহৃত হওয়া শব্দ:
  static_assert, constexpr
- পূর্ব প্রক্রিয়কের (preprocessor) জন্য শব্দ:
  if, elif, else, endif, defined, ifdef, ifndef, define, undef,
  include, line, error, pragma
- বিভিন্ন আকারের অক্ষরের জন্য শব্দ: char16_t, char32_t, wchar_t
- বিবিধ শব্দ: alignas, alignof, asm, concept, requires, thread_local

৩.৫ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. শনাক্তক (identifier) কী? ক্রমলেখতে শনাক্তকের ভুমিকা কী?

২. সিপিপিতে শনাক্তকের (identifier) নাম করণের নিয়মাবলী বর্ণনা করো।

৩. ক্রমলেখতে (program) গঠনগত ভাবে অবৈধ নাম ব্যবহার করলে কী ঘটে?

৪. অর্থবোধক নাম কী? ক্রমলেখতে অর্থবোধক নাম ব্যবহার করা উচিৎ কেন?

৫. সিপিপি একটি লিপি সংবেদনশীল (case sensitive) ভাষা, এর মানে কী ?

৬. সংরক্ষিত ও চাবি শব্দ কী? এগুলো কেন শনাক্তক হিসাবে ব্যবহার করা যায় না?

চর্চামুলক প্রশ্ন: নীচের শব্দগুলো গঠনগত (syntactically) ভাবে শনাক্তকের (identifier) নাম হিসাবে বৈধ নাকি অবৈধ? যদি বৈধ হয় তাহলে অর্থবোধক (meaningful) নাকি অর্থবোধক নয়? অথবা কোন শব্দ কি সংরক্ষিত বা চাবি শব্দ (reserved or key word)? প্রথমে নিজে নিজে উত্তর বের করার চেষ্টা করবে, একান্ত না পারলে নীচের সমাধান দেখবে।

১. void
২. MAX-ENTRIES
৩. double
৪. time
৫. G
৬. Sue's
৭. return
৮. cout
৯. xyz123
১০. part#2
১১. "char"
১২. #include
১৩. a_long-one
১৪. _xyz
১৫. 9xyz
১৬. main
১৭. mutable
১৮. max?out
১৯. Name
২০. name
২১. name_1
২২. Int
২৩. INT
২৪. _SUM

২৫.	sum_of_numbers	২৮.	printf	৩১.	$sum
২৬.	firstName	২৯.	int	৩২.	num^2
২৭.	Identifier	৩০.	pow	৩৩.	num 1

চর্চামুলক উত্তর: উপরের প্রশ্নগুলোর উত্তর এখানে দেয়া হচ্ছে। প্রথমে নিজে নিজে উত্তর বের করার চেষ্টা করবে, একান্ত না পারলে এই সমাধান দেখবে।

১. void : সংরক্ষিত শব্দ, কোন প্রকারেরই না এমন বুঝানো হয়

২. MAX-ENTRIES : বৈধ শনাক্তক, অর্থবোধক

৩. double : সংরক্ষিত শব্দ, বড় আকারের ভগ্নকের জন্য

৪. time : বৈধ শনাক্তক, কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

৫. G : বৈধ শনাক্তক, কিন্তু অর্থ বুঝা যাচ্ছে না, যদি না পারিপার্শ্বিকতা পরিস্কার থাকে

৬. Sue's : অবৈধ শনাক্তক কারণ নামে ' ব্যবহার করা যায় না

৭. return : সংরক্ষিত শব্দ, বিপাতক থেকে ফেরত গমন

৮. cout : বৈধ শনাক্তক, কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

৯. xyz123 : বৈধ শনাক্তক, কিন্তু অর্থবোধক কিনা পরিস্কার না

১০. part#2 : অবৈধ শনাক্তক কারণ নামে # ব্যবহার করা যায় না

১১. "char" : অবৈধ শনাক্তক কারণ নামে " ব্যবহার করা যায় না

১২. #include : পূর্ব-প্রক্রিয়কের (preprocessor) জন্য সংরক্ষিত শব্দ

১৩. a_long-one : বৈধ শনাক্তক, কিন্তু অর্থ সেই ভাবে পরিস্কার নয়।

১৪. _xyz : বৈধ শনাক্তক, কিন্তু অর্থ সেই ভাবে পরিস্কার নয়

১৫. 9xyz : অবৈধ শনাক্তক, নামের শুরুতে অঙ্ক থাকতে পারে না, পরে থাকতে পারে

১৬. main : সংরক্ষিত শব্দ নয়, কিন্তু পরিত্যাজ্য কারণ এটি প্রত্যেক ক্রমলেখতেই থাকে

১৭. mutable : সংরক্ষিত শব্দ, কোন ধ্রুবক ও বিশেষ অবস্থায় পরিবর্তন যোগ্য হলে

১৮. max?out : অবৈধ শনাক্তক, নামে ? চিহ্ন থাকতে পারবে না

১৯. Name : বৈধ শনাক্তক, অর্থবোধক, কীসের নাম সেটা পরিস্কার নয়

২০. name : বৈধ শনাক্তক, অর্থবোধক, কীসের নাম সেটা পরিস্কার নয়

২১. name_1 : বৈধ শনাক্তক, অর্থবোধক, কীসের নাম সেটা পরিস্কার নয়

২২. Int : বৈধ শনাক্তক, তবে সংরক্ষিত শব্দ int এর সাথে বিভ্রান্তি দেখা দিতে পারে

২৩. INT : বৈধ শনাক্তক, তবে সংরক্ষিত শব্দ int এর সাথে বিভ্রান্তি দেখা দিতে পারে

২৪. _SUM : বৈধ শনাক্তক, অর্থবোধক, যোগফলের জন্য

২৫. sum_of_numbers : বৈধ শনাক্তক, অর্থবোধক

২৬. firstName : বৈধ শনাক্তক, অর্থবোধক, অনেকের পছন্দের

২৭. Identifier : বৈধ শনাক্তক, অর্থবোধক, কীসের শনাক্তক পরিস্কার নয়

২৮. printf : বৈধ শনাক্তক, অর্থবোধক, কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

২৯. int : সংরক্ষিত শব্দ, পূর্ণক উপাত্ত ধারনের জন্য উপাত্ত প্রকরণ

৩০. pow : বৈধ শনাক্তক, অর্থবোধক, কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

৩১. $sum : অবৈধ শনাক্তক, নামে $ চিহ্ন ব্যবহার করা যায় না

৩২. num^2 : অবৈধ শনাক্তক, নামে ^ চিহ্ন ব্যবহার করা যায় না

৩৩. num 1 : অবৈধ শনাক্তক, নামে ফাঁকা ব্যবহার করা যায় না

৩.৬ গণনা পরিভাষা (Computing Terminologies)

- শনাক্তক (identifier)
- নিম্নদাগ (underscore)
- সাজন (array)
- সংগঠিত (structured)
- পরিগণনা (programming)
- সংগঠিত পরিগণনায় (structured programming)
- বস্তুমুখী (object-oriented)
- বস্তুমুখী পরিগণনা (object oriented programming)
- ত্রুটি সামলানো (error handling)
- যুক্তি অণুক্রিয়া (logical operators)
- বিটপ্রতি অণুক্রিয়া (bit-wise operators)
- উপাত্ত প্রকরণ (data type)
- ছাঁচ (template)
- সংকলন সময়ে (compile-time)

অধ্যায় ৪

যোগান ও আরোপণ (Input and Assignment)

ক্রমলেখতে (program) উপাত্ত (data) কোথা থেকে আসে? হয় আমরা ক্রমলেখয়ের ভিতরে সরাসরি লিখে দেই, যেমনটি আগের পাঠগুলোতে করেছি, আর না হয় আমরা উপাত্ত ব্যবহারকারী-দের কাছে থেকে যোগান (input) নেই। উপাত্ত যোগান নিয়ে সেটিকে ধারন করার উদ্দেশ্যে আমরা চলকে (variable) আরোপণ (assign) করি যাতে ওই উপাত্ত পরে কাজে লাগানো যায়।

৪.১ উপাত্ত যোগান (Data Input)

সিপিপিতে এমন একটি ক্রমলেখ (program) লিখো যেটি যে কোন আয়তের ক্ষেত্রফল ও পরিসী-মা নির্ণয় করতে পারে। তোমার ক্রমলেখ তুমি মাত্র একবারই সংকলন (compile) করতে পারবে, আর প্রত্যেক আলাদা আয়তের জন্য তুমি ক্রমলেখটি বারবার কেবল চালাতে পারবে, কিন্তু ক্রম-লেখয়ের ভিতরে দৈর্ঘ্য ও প্রস্থ বদলে দিয়ে বারবার সংকলন করতে পারবে না। তারমানে তোমাকে দৈর্ঘ্য ও প্রস্থ যোগান (input) হিসাবে তোমার ক্রমলেখ ব্যবহারকারীর কাছে থেকে নিতে হবে।

উক্ত ক্রমলেখ লেখার আগে চলো আমরা কিছু দরকারী আলোচনা সারি। আমরা যখন কোন গণনা সমস্যার (computing problem) সমাধান করতে চাই, যেমন আলোচ্য ক্ষেত্রে আমরা আয়তের দৈর্ঘ্য ও প্রস্থ জেনে তার ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে চাই, তখন আমরা মুলত একটি ক্রমলেখ (program) ব্যবহার করবো, মানে আমরা ক্রমলেখটি চালাবো (run)। এখন এই ক্রমলেখ হয়তো আমরা নিজেরা তৈরী করবো অথবা অন্য কেউ আমাদের তৈরী করে দিবে। বেশীর ভাগ ক্ষেত্রে ক্রমলেখটি অন্যের তৈরী করা দেয়া, আমরা কেবল ব্যবহারকারী।

ভেবে দেখো ক্রমলেখ তৈরী করা (write) আর চালানো (run) আসলে দুটো ভিন্ন ঘটনা। এই দুটো ঘটনা পরপর একসাথে ঘটবে এরকম সবসময় হয় না। বরং বেশীর ভাগ সময়ে এই ঘটনা দুটো আসলে ভিন্ন দুটি স্থানে ভিন্ন দুটি সময়ে ভিন্ন দুই ব্যক্তির দ্বারা সংঘটিত হয়। তাছাড়া ক্রমলেখ যে চালাবে সে হয়তো কেবল একটা আয়তের ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে চায় না বরং তার হাতে হয়তো অনেক অনেক আয়ত আছে আর সে সবগুলো আয়তের জন্যই ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে চায়। সুতরাং প্রতিটা আয়তের জন্য তার একটা করে আলাদা ক্রমলেখ লাগবে যদি ক্রমলেখয়ের ভিতরে আয়তের দৈর্ঘ্য ও প্রস্থ দিয়ে দেয়া হয়। অথবা তার এমন একটা ক্রমলেখ লাগবে যেটা কোন না কোন ভাবে সবগুলো আয়তের জন্যই কাজ করবে, আর সঠিক ভাবেই করবে অর্থাৎ ক্রমলেখটি মুলত সুত্রের (formula) ওপর নজর দেবে, উপাত্তের (data) ওপর নয়।

৪.১. উপাত্ত যোগান (Data Input)

আমরা উপরে যেসব অবস্থা আলোচনা করলাম সেই সব অবস্থায় ক্রমলেখক (programmer) ক্রমলেখ তৈরী করার সময় জানবেন না আয়তের দৈর্ঘ্য ও প্রস্থ কী হবে, সেটি জানা সম্ভব হবে পরে ব্যবহারকারী যখন ক্রমলেখটি চালাবেন কেবল তখন। প্রশ্ন হচ্ছে এমতাবস্থায় ক্রমলেখক উপাত্ত (data) ছাড়া কী ভাবে ক্রমলেখ তৈরী করবেন। সত্যি বলতে উত্তর তো গণিতেই আছে চলক (variable) ব্যবহার করে। আর আমরা তো ইত্যমধ্যে ক্রমলেখতে চলক ব্যবহার করেছিই। আমাদের কেবল যেটা করা দরকার তা হলো ক্রমলেখয়ের ভিতরে দৈর্ঘ্য বা প্রস্থ সরাসরি লিখে না দিয়ে ওইটা যাতে ব্যবহারকারী ক্রমলেখ চালানোর সময় দিয়ে দিতে পারে সেই ব্যবস্থা করা। নীচের ক্রমলেখতে আমরা তাই করেছি। আমরা ব্যবহারকারীর কাছে থেকে চলকের মান উপাত্ত হিসাবে যোগান (input) নিয়েছি। এবার আমরা ওই ক্রমলেখটির সংশ্লিষ্ট অংশটুকু বিশ্লেষণ করি।

ফিরিস্তি ৪.১: উপাত্ত যোগানের ক্রমলেখ (Programs with Data Input)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int doirgho;  // আয়তের দৈর্ঘ্যের জন্য চলক
  cin >> doirgho; // দৈর্ঘ্য যোগান হিসাবে নেওয়া হবে

  int prostho;  // আয়তের প্রস্থের জন্য চলক
  cin >> prostho; // প্রস্থ যোগান হিসাবে নেওয়া হবে

  // ক্ষেত্রফল ও পরিসীমার সুত্র
  int khetrofol = doirgho * prostho;
  int porishima = 2*(doirgho + prostho);

  //  ক্ষেত্রফল ও পরিসীমা ফলন
  cout <<"khetrofol holo "<< khetrofol << endl;
  cout <<"porishima holo "<< porishima << endl;

  return EXIT_SUCCESS;  // সফল ভাবে ফেরত
}
```

যোগান-ফলন (input-output)

```
13
12
khetrofol holo 156
porishima holo 50
```

উপরের ক্রমলেখতে খেয়াল করো আমরা দৈর্ঘ্যের জন্য একটি চলক doirgho ঘোষনা করেছি, কিন্তু সাথে সাথে তার কোন আদিমান আরোপ (initial value assign) করি নাই। কারণ আগেই যেমন আলোচনা করলাম, আমরা যখন ক্রমলেখ লিখছি তখন আমরা আসলে জানিনা যে doirgho এর মান কতো। আমরা বরং ওইটা ব্যবহারকারীর কাছে থেকে নেবো। আর সে কারণে আমরা cin >> doirgho; লিখেছি। এখানে cin হল console in। সাধারণত যোগান যন্ত্র (input device) চাপনি (keyboard) ও টিপনি (mouse) আর ফলন যন্ত্র (output device) নজরি (monitor) মিলিয়ে হল আমাদের console বা যন্ত্রালয়। তো console in বলতে আমরা এখানে যোগান যন্ত্র বিশেষ করে চাপনি (keyboard) থেকে যোগান (input) নেয়া বুঝাচ্ছি। তাহলে cin ব্যবহারকারীর কাছে থেকে চাপনির মাধ্যমে সংখ্যাটা নিয়ে সেটা doirgho চলকের ভিতরে দিয়ে দিবে। এতে ওই চলকে একরকমের মান আরোপণ (value assign) হয়ে যাবে।

ব্যবহারকারীর কাছে থেকে দৈর্ঘ্য নেবার পরে আমাদের প্রস্থও নিতে হবে। উপরের ক্রমলেখতে খেয়াল করো আমরা দৈর্ঘ্যের মতো করেই প্রস্থের জন্যও prostho নামে একটা int ধরনের চলক ঘোষনা করেছি আর তার পরের সারিতে cin ব্যবহার করে prostho এর মান ব্যবহারকারীর কাছে থেকে নেয়ার কথা লিখেছি। উপরের ক্রমলেখয়ের বাঁকী অংশটুকু তো আগের পাঠের ক্রমলেখগুলোতে যেমন দৈর্ঘ্য ও প্রস্থ ব্যবহার করে ক্ষেত্রফল ও পরিসীমার সুত্র লিখা হয়েছে আর তারপরে ফলন (output) দেখানো হয়েছে ঠিক তেমনই। আমরা সেগুলো আর আলোচনা করছি না।

এবার আমরা আর একটু আলোচনা করি উপরের ক্রমলেখটি সংকলন (compile) করে চালালে কী ঘটবে তা নিয়ে। উপরের ক্রমলেখটি চালালে আমরা দেখব পর্দায় (screen) কিছু আসছে না, চটুলটা (cursor) কেবল লাফালাফি করছে। আমরা এই অবস্থায় দৈর্ঘ্যের মান, ধরা যাক 13 চেপে ভুক্তি (enter) চাপবো। ভিতরে ভিতরে cin ওই মান নিয়ে doirgho চলকের মধ্যে রেখে দিবে। চটুলটা (cursor) তারপরও লাফালাফি করবে। আমরা তখন 12 দিয়ে ভুক্তি (enter) চাপবো, cin ওইটা prostho চলকে রেখে দিবে। তারপর পর্দায় আমরা ফলন দেখতে পাবো। প্রথম সারিতে থাকবে khetrofol holo 156 আর পরের সারিতে porishima holo 50।

উপরে ক্রমলেখতে আমরা চাইলে কিছু সংক্ষিপ্তকরণ করতে পারি। যেমন দৈর্ঘ্য ও প্রস্থ ঘোষনা (declaration) ও যোগান (input) নেয়া চার সারিতে না করে আমরা ওগুলোকে মাত্র দুই সারিতে সারতে পারি। প্রথম সারিতে আমরা চলক দুটো ঘোষনা করবো। আর পরের সারিতে আমরা চলক দুটোর যোগান নিবো। নীচের ক্রমলেখাংশে (program segment) এইগুলো দেখানো হলো।

```
int doirgho, prostho;    // আয়তের দৈর্ঘ্যে ও প্রস্থের জন্য চলক
cin >> doirgho >> prostho;  // দৈর্ঘ্য ও প্রস্থ যোগান নেওয়া হবে
```

আর সেক্ষেত্রে ক্রমলেখটি চালানোর সময় যোগান নেয়ার অংশ নিম্নরূপ হবে। লক্ষ্য করবে চটুল (cursor) যখন যোগান নেবার জন্য লাফাতে থাকবে, আমরা তখন 13 ও 12 সংখ্যা দুটি ফাঁকা দিয়ে এক সাথে দিয়েই ভুক্তি (enter) চাপতে পারবো, অথবা চাইলে 13 লিখে ভুক্তি চেপে তারপর 12 লিখে আবার ভুক্তি চাপতে পারবো। আর ফলনের অংশ আগের মতোই হবে।

```
13 12
```

কেউ যদি চায় তাহলে কিন্তু ফলন অংশেও এরকম সংক্ষিপ্তকরণ করতে পারে। যেমন ক্ষেত্রফল ও পরিসীমা চাইলে এক সারিতেই ফলন দিতে পারে।

```
cout << "khetrofol ar porishima holo " << khetrofol
 << " " << porishima << endl; // cout হতে এই পর্যন্ত
 পুরোটা আসলে এক সারিতে
```

তবে সবকিছু একবার cout দিয়ে দেওয়ার চেয়ে আমরা হয়তো দুইবারে দিতে চাইবো।

```
cout << "khetrofol ar porishima holo ";
cout << khetrofol << " " << porishima << endl;
```

উপরের উভয় ক্ষেত্রে পর্দায় ফলন কিন্তু একসারিতেই আসবে।

```
khetrofol ar porishima holo 156 50
```

৪.২ যোগান যাচনা (Input Prompt)

সিপিপিতে এমন একটি ক্রমলেখ (program) রচনা করো যেটি যে কোন আয়তের ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে পারে। তোমার ক্রমলেখ আয়তের দৈর্ঘ্য ও প্রস্থ ব্যবহারকারীর কাছে থেকে যোগান (input) নিবে। আর দৈর্ঘ্য ও প্রস্থ যোগান নেবার আগে তোমার ক্রমলেখ অবশ্যই ব্যবহারকারীকে দৈর্ঘ্য ও প্রস্থের মান জিজ্ঞেস করবে অর্থাৎ যাচনা (prompt) করবে।

ফিরিস্তি ৪.২: যোগান যাচনার ক্রমলেখ (Program with Input Prompt)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int doirgho;          // আয়তের দৈর্ঘ্যের জন্য চলক
  cout << "doirgho koto? ";    // মান যাচনা করা হচ্ছে
  cin >> doirgho;         // দৈর্ঘ্য যোগান হিসাবে নেওয়া হবে

  int prostho;          // আয়তের প্রস্থের জন্য চলক
  cout << "prostho koto? "; // মান যাচনা করা হচ্ছে
  cin >> prostho;    // প্রস্থ যোগান হিসাবে নেওয়া হবে

  // ক্ষেত্রফল ও পরিসীমার সুত্র
  int khetrofol = doirgho * prostho;
  int porishima = 2*(doirgho + prostho);

  //  ক্ষেত্রফল ও পরিসীমা ফলন দেয়া হবে
  cout <<"khetrofol holo "<< porishima << endl;
  cout <<"porishima holo "<< porishima << endl;

  return EXIT_SUCCESS;  // সফল ভাবে ফেরত
}
```

যোগান-ফলন (input-output)

```
doirgho koto? 13
prostho koto? 12
khetrofol holo 156
porishima holo 50
```

আগের পাঠের ক্রমলেখতে আমরা চলকের মান ব্যবহারকারীর কাছে থেকে নেয়ার জন্য cin ব্যবহার করেছি। ওই ক্রমলেখটি যখন আমরা চালাই তখন দেখি পর্দায় (screen) কিছু নাই আর চটুলটা (cursor) কেনো যেনো লাফালাফি করছে। সেই অবস্থায় আমরা প্রথমে দৈর্ঘ্যের মান 13 দিয়ে ভুক্তি (enter) চেপেছি। চটুলটা তারপরও লাফালাফি করছিল। আমরা তখন 12 দিয়ে ভুক্তি চেপেছি। তারপর পর্দায় ফলন এসেছিল প্রথম সারিতে khetrofol holo 156 আর পরের সারিতে porishima holo 50। তো এই যে চটুলটা (cursor) লাফালাফি করছিল দৈর্ঘ্য ও প্রস্থের মান নেয়ার জন্য এইটা আমরা বুঝতে পারি কারণ আমরা নিজেরাই এক্ষেত্রে ক্রমলেখটি তৈরী write) করেছি আর নিজেরাই সেটা সংকলন (compile) করে চালাচ্ছি (run)। আমরা এক্ষেত্রে জানি যে আমাদের ক্রমলেখটি প্রথমে দৈর্ঘ্য চাচ্ছে আর সেটা দেবার পর প্রস্থ চাচ্ছে। এবার ভেবে দেখো আমাদের লেখা ক্রমলেখ যদি আমরা ছাড়া অন্য কেউ চালায় (run) তাহলে সে কী ভাবে জানবে চটুলটি (cursor) ওই অবস্থায় কেন লাফাচ্ছে। সে কি আসলেই দৈর্ঘ্য বা প্রস্থ নেয়ার জন্য অপেক্ষা করছে নাকি ভিতরে ভিতরে ঘটনা অন্য কিছু, সে হয়তো অন্য কিছু করছে।

তো ওপরের সমস্যা সমাধানের জন্য আমরা যেটি করবো সেটি হলো আমাদের ক্রমলেখতে cin >> doirgho; লেখার আগে আমরা একটা বার্তা দেখাবো যে আমরা দৈর্ঘ্যের মান চাই। উপরের ক্রমলেখ খেয়াল করো cin >> doirgho; লেখার আগে আমরা cout << "doirgho koto? "; লিখে আসলে সেটাই করতে চাইছি। এই ক্রমলেখ যখন চালানো হবে তখন প্রথমে পর্দায় doirgho koto? দেখা যাবে। আর cout এর শেষে আমরা যেহেতু endl অর্থাৎ end line দেই নাই, চটুলটা (cursor) সেহেতু ওই একই সারিতে লাফাইতে থাকবে, লাফাইতে থাকবে মুলত cin >> doirgho; এর কারণে doirgho এর মান নেয়ার জন্য। আমরা তখন doirgho এর মান দিয়ে ভুক্তি (enter) চাপবো। তাহলে "চটুল কেন লাফায়?" আমরা এই সমস্যার সমাধান করে ফেললাম কেমন! এই যে যোগান (input) নেবার আগে একটা বার্তা দিয়ে ব্যবহারকারীকে জানানো যে আমরা কী যোগান চাই, এই ব্যাপারটিকে বলা হয় যোগান যাচনা (input prompt). উপরের ক্রমলেখতে খেয়াল করো আমরা প্রস্থের জন্যেও একই ভাবে যোগান (input) নেবার আগে "prostho koto? " বার্তা দিয়ে যোগান যাচনা (input prompt) করেছি। তাহলে এখন থেকে তোমার ক্রমলেখতে যোগান নেবার আগে অবশ্যই যোগান যাচনা করবে, কেমন?

উপরে ক্রমলেখতে আমরা চাইলে কিছু সংক্ষিপ্তকরণ করতে পারি। যেমন দৈর্ঘ্য ও প্রস্থ ঘোষনা (declaration), যোগান যাচনা (input prompt) করা, ও যোগান (input) নেয়া ছয় সারিতে না করে আমরা ওগুলোক মাত্র তিন সারিতে সারতে পারি। প্রথম সারিতে আমরা চলক দুটো ঘোষনা করবো। আর পরের সারিতে আমরা যোগান যাচনা করবো তারপরে সারিতে চলক দুটোর মান যোগান নিবো। নীচে ক্রমলেখাংশে (program segment) এইগুলো দেখানো হলো।

```
int doirgho, prostho;      // দৈর্ঘ্যে ও প্রস্থের জন্য চলক
cout << "doirgho o prostho koto? ";// একসাথে যাচনা
cin >> doirgho >> prostho;       // দৈর্ঘ্য ও প্রস্থ যোগান
```

আর সেক্ষেত্রে ক্রমলেখটি চালানোর সময় যোগান নেয়ার অংশ নিম্নরূপ হবে। অর্থাৎ ক্রমলেখ চালালে doirgho o prostho koto? দেখানোর পরে চটুলটা (cursor) যোগান নেবার জন্য লাফাতে থাকবে। আমরা 13 ও 12 সংখ্যা দুটি ফাঁকা দিয়ে এক সাথে দিয়েই ভুক্তি (enter) চাপতে

পারবো, অথবা চাইলে 13 লিখে ভুক্তি চেপে তারপর 12 লিখে আবার ভুক্তি চাপতে পারবো। আর ফলনের অংশ আগের মতোই হবে, কাজেই আমরা সেটা আর দেখাচ্ছি না।

```
doirgho o prostho koto? 13 12
```

৪.৩ মান আরোপণ (Value Assignment)

ক্রমলেখতে (program) চলক নিয়ে তাতে মান আরোপণ (value assign) করলে আসলে কী ঘটে? চলকে একটা মান আগে থেকে আছেই, এমতাবস্থায় আরেকটা মান আরোপ করলে কী ঘটে? একটা চলক থেকে আরেকটা চলকে মান আরোপ করলেই বা কী ঘটে?

```
int amar;
int tomar = 5;
```

উপরে আমরা দুটো চলক ঘোষনা (variable declare) করলাম: একটার নাম amar আর আরেকটার নাম tomar, দুটোই int ধরনের অর্থাৎ পূর্ণক, একটাতে আদিমান (initial value) দিয়ে দিলাম আর একটাতে দিলাম না। আমরা যখন চলক ঘোষনা করি তখন আসলে আমরা গণনির (computer) স্মরনিতে (memory) কিছু জায়গা দখল করি। ধরে নিতে পারো স্মরনি হল একটা রাস্তার পাশে অনেকগুলো একই রকম বাড়ী। কোন চলক ঘোষনা করার সময় আমরা আসলে ওই বাড়ীগুলোর একটা দখল করে সেই বাড়ীটার নাম দিয়ে দেই আমাদের চলকের নামে। তোমরা নিশ্চয় দেখেছো অনেকেরই বাড়ীর নাম থাকে যেমন "শান্তি নীড়"। আমাদের চলক বাড়ীগুলোর নাম amar ও tomar। তো আমরা যখন উপরের দুটো চলক ঘোষনা করলাম তখন স্মরনিতে ওই রকম দুটো জায়গা নিয়ে তাদের নাম দিয়ে দিলাম amar আর tomar। এখন কথা হচ্ছে স্মরণিতে (memory) ওই জায়গায় আমরা আসলে রাখবো কী? উত্তরটাতো সহজ আমরা রাখবো চলকটির মান। যখন আমরা আদিমান দিয়ে দিলাম তখন ওই জায়গাতে আমাদের দেয়া মানটা থাকবে, আর যখন আদিমান দিবো না, তখনও ওই জায়গাটিতে আগে থেকে যাই ছিল তাই থাকবে।

```
amar = tomar;
```

এবার আমরা যদি উপরের মতো করে tomar এর মান amar এ আরোপ করি তাহলে কী ঘটবে? আসলে উপরের এই বিবৃতি (statement) চালানোর পরে amar এর আগের মান মুছে গিয়ে সেটার নতুন মান হয়ে যাবে tomar এর মানের সমান অর্থাৎ amar এর মানও হবে 5। এখানে একটা গুরুত্বপূর্ণ বিষয় বলে রাখতে হবে যে এই যে tomar থেকে amar এ মান আরোপ করা হলো এতে কিন্তু tomar এর মানে কোন পরিবর্তন হবে না। অর্থাৎ tomar এর মান আগের মতো 5-ই থাকবে। আরোপণে (assignment) সমান চিহ্নের বামে যা থাকে সেটাকে লক্ষ্য (target) আর ডানে যেটা থাকে সেটাকে (source) বলা হয়, কারণ উৎস থেকে মান নিয়ে লক্ষ্যে আরোপ করা হয়। উপরের আরোপণে amar = চিহ্নের বামে তাই এটি লক্ষ্য আর tomar ডানপাশে তাই এটি উৎস। আরোপণের ফলে লক্ষ্যের মান বদলে কিন্তু উৎসের মান বদলে না, একই থাকে।

উপরের ক্রমলেখাংশ (program segment) আর ফলনাংশ (output segment) লক্ষ্য করো। আমরা প্রথমে চলক x ঘোষনা করে তার আদি মান (initial value) দিয়েছি 3, তারপর চলক y ঘোষনা করে তার আদিমান দিয়েছি x+5 অর্থাৎ 3 + 5 = 8। এই পর্যায়ে ফলন দেখানো হয়েছে x আর y দুটোর মানেরই। ফলনাংশে আমরা দেখতে পাচ্ছি `x 3 y 8`। তারপর ক্রমলেখাংশে আমরা লিখেছি `x = y * 3;` ফলে x এর মান হবে এখন 8 * 3 = 24, আর y এর মান কিন্তু একই থাকবে, কারণ y এর মান আমরা কেবল ব্যবহার করেছি, y এ তো কোন মান আরোপ করি

নাই। ক্রমলেখাংশে পরের বিবৃতিতে (statement) আমরা x ও y এর তখনকার মান দেখিয়েছি, ফলনাংশে সেটা ঠিকই x 24 y 8 দেখাচ্ছে। ক্রমলেখাংশে এরপরের বাক্যে আমরা আবার x এ মান আরোপ করেছি x = y + 3 * 4; তো এর ফলে আগের মতোই y এর মান বদল হবে না, কিন্তু x এর নতুন মান হয়ে যাবে 8 + 3 * 4 = 20, যা পরের cout এর মাধ্যমে ফলনাংশে ঠিকই দেখানো হয়েছে x 20 y 8। ক্রমলেখাংশে এরপর আমরা দেখি y = x * 2; এর ফলে y এর নতুন মান হবে y = 20 * 2 = 40, আর x এর মান এবার আগে যা ছিলো তাই থাকবে, কারণ x এর মান কেবল ব্যবহার করা হয়েছে, x এ কোন মান আরোপ করা হয় নি।

ক্রমলেখাংশ (program segment)

```
int x = 3;                  // আদি মান আরোপ
int y = x + 5;              // আদি মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও

x = y * 3;                  // পুনরায় মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও

x = y + 3 * 4;              // পুনরায় মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও

y = x * 2;                  // পুনরায় মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও
```

ফলনাংশ (output segment)

```
x 3 y 8
x 24 y 8
x 20 y 8
x 20 y 40
```

সবমিলিয়ে একটা বিষয় দেখা যাচ্ছে আরোপণে = চিহ্নের বামে থাকা লক্ষ্য চলকের (target variable) মান কেবল পরিবর্তন হয়, আর = চিহ্নের ডানে থাকা চলক (variable) বা রাশির (expression) এর মান কোন পরিবর্তন হয় না। আরেকটি ব্যাপার হলো কোন চলকে পরে কোন নতুন মান আরোপ না হওয়া পর্যন্ত আগেরবার যে মান আরোপ করা হয়েছিল সেটাই থাকে।

8.8 মান অদল-বদল (Value Swapping)

ধরো তোমার দুটো চলক (variable) আছে x আর y আর তাদের মান যথাক্রমে 12 ও 13। তো তোমাকে এমন কিছু বিবৃতি (statement) লিখতে হবে যাতে ওই বিবৃতিগুলো চালানোর (run) পরে আমরা x আর y এর মান যথাক্রমে 13 আর 12 পাই অর্থাৎ মানদুটো অদল-বদল হয়ে যায়।

```
int x = 12; // x এর মান আরোপ করা হলো
int y = 13; // y এর মান আরোপ করা হলো
```

উপরে আমরা কেবল চলক x আর y ঘোষনা করে তাদের আদিমান হিসাবে 12 ও 13 দিয়ে দিলাম। এখন আমরা এমন কিছু করবো যাতে x আর y মান অদল-বদল হয়ে যায়। প্রথমেই আমরা

একটা চটুল সমাধান করি। তোমাদের মধ্যে যারা দুষ্টু ধরনের আর চটপটে তারা সাধারণত এই সমাধানটি করতে চাইবে। নীচের বিবৃতি দুটো লক্ষ্য করো: আমরা স্রেফ x এর মধ্যে সরাসরি 13 আরোপ করেছি আর y এর মধ্যে 12 আরোপ করেছি। ব্যস হয়ে গেল x আর y এর মান অদল-বদল! আসলে আমরা কী এইটে চেয়েছিলাম? এখানে তো চলক দুটোর মধ্যে একটা থেকে আরেকটাতে মান নেয়ার মতো কোন ঘটনা ঘটে নি, কাজেই কোন অদল বদলের কিছু ঘটে নি!

```
x = 13; // x এর মান আরোপ করা হলো
y = 12; // y এর মান আরোপ করা হলো
```

অদল-বদল বুঝার জন্য চিন্তা করো তোমার দুটি পেয়ালা আছে: কাঁচের পেয়ালা আর কাঁসার পেয়ালা। কাঁচের পেয়ালায় আছে আঙুরের রস আর কাঁসার পেয়ালায় কমলার রস। এখন তুমি এই পেয়ালা দুটোতে থাকা ফলের রস অদল-বদল করতে চাও যাতে কাঁচের পেয়ালায় থাকে কমলার রস আর কাঁসার পেয়ালায় থাকে আঙুরের রস। তো এখন তুমি কী করবে। তুমি তো আর সরাসরি এক-টার ফলের রস আরেকটাতে ঢেলে দিতে পারো না। তোমাকে যেটা করতে হবে তা হলো আরেকটা পেয়ালা নেয়া। ধরো সেটা কাঠের পেয়ালা। এই কাঠের পেয়ালাটি তুমি একটা থেকে আরেকটাতে ঢালাঢালির কাজে ব্যবহার করবে। তাহলে এই অতিরিক্ত কাঠের পেয়ালা কাজে লাগিয়ে কীভাবে তোমার কাঁচ আর কাঁসার পেয়ালার ফলের রস অদল-বদল করা যায়, আমরা নীচে তা দেখি।

১. একদম শুরুতে কাঁচের পেয়ালায় রয়েছে আমাদের আঙুরের রস আর কাঠের পেয়ালা খালি। সুতরাং কাঁচের পেয়ালা থেকে আঙুরের রস কাঠের পেয়ালায় ঢালো। ফলে কাঠের পেয়ালায় থাকলো আঙুরের রস আর কাঁচের পেয়ালা খালি হয়ে গেলো।

২. কাঁচের পেয়ালা যেহেতু এখন খালি আর কাঁসার পেয়ালায় আছে কমলার রস, আমরা তাই কাঁসার পেয়ালার কমলার রস কাঁচের পেয়ালায় ঢালবো। ফলে কাঁচের পেয়ালায় থাকলো কমলার রস আর কাঁসার পেয়ালা খালি হয়ে গেলো।

৩. কাঁসার পেয়ালা যেহেতু এখন খালি আর কাঠের পেয়লায় আছে আঙুরের রস, আমরা তাই কাঠের পেয়ালার আঙুরের রস কাঁসার পেয়ালায় ঢালবো। ফলে কাঁসার পেয়ালায় থাকলো আঙুরের রস আর কাঠের পেয়ালা খালি হয়ে গেলো।

উপরের ধাপ তিনটি সম্পন্ন করলেই আমাদের এক পেয়ালার ফলের রস আরেক পেয়ালায় অদল-বদল হয়ে যাবে। তো পেয়ালা দুটোর রস অদল-বদলের মতোই আসলে আমাদের চলকদু-টোর মান অদল-বদল করতে হবে। একটা অতিরিক্ত পেয়ালার মতো আমাদের এখানেও লাগবে একটা অতিরিক্ত চলক। ধরে নেই আমাদের সেই অতিরিক্ত চলক হলো z। আমরা তাহলে এই অতিরিক্ত চলক কাজে লাগিয়ে আমাদের x আর y চলকের মান অদল-বদল করে ফেলি।

```
z = x;  // z হলো 12 আর x আছে 12, y আছে 13
x = y;  // x হলো 13 আর y আছে 13, z আছে 12
y = z;  // y হলো 12 আর z আছে 12, x আছে 13
```

তো উপরের তিনটি বিবৃতি চালালেই আমাদের x আর y চলক দুটোর মান অদল-বদল হয়ে গেলো। খেয়াল করেছো চলকের নাম যখন আমরা আরোপ চিহ্নের ডানে লিখেছি তখন আসলে আমরা চলকটির ডান-মান বুঝিয়েছি, আর যখন বামে লিখেছি তখন বাম-মান বুঝিয়েছি। তবে পে-য়ালা আর ফলের রসের অদল বদলের সাথে চলক আর মানের অদল-বদলের কিন্তু কিছুটা তফাৎ আছে। তফাৎটা হলো ফলের রস এক পেয়ালা থেকে আরেক পেয়ালয় ঢাললে যেটা থেকে ঢালা হলো সেই পেয়ালা খালি হয়ে যায়। কিন্তু চলকের ক্ষেত্রে z = x; করলে চলক x এর মান চলক z

এ আরোপ হয় ঠিকই, কিন্তু চলক x কিছুতেই খালি হয় না, বরং তার যে মান ছিলো সেটাই থাকে। চলকের মান বদলে যায় কেবল যখন এতে নতুন মান আরোপ করা হয়।

৪.৫ আরোপণের বাম ও ডান (Assignment Left and Right)

কোন চলকের (variable) বাম-মান (lvalue) ও ডান-মান (rvalue) বলতে কী বুঝো? কোন চলকে মান আরোপণ করতে গেলে আমরা আরোপ (assign) = চিহ্ন দিয়ে বামে ও ডানে কিছু লিখি যেমন y = x;। এখানে বামেরটিকে বলা হয় লক্ষ্য (target) আর ডানেরটিকে বলা হয় উৎস (source)। আরোপণের ফলে ডান পাশের উৎস থেকে মান বাম পাশের লক্ষ্যে আরোপিত হয়। কথা হচ্ছে আরোপ = চিহ্নের বামে আমরা কী কী দিতে পারবো বা পারবো না, আর ডানেই বা কী কী দিতে পারবো বা পারবো না? তাছাড়া একটা চলকের নাম আরোপ = চিহ্নের বাম বা ডানপাশে লিখলে এই দুই ক্ষেত্রে চলকের ভুমিকায় আসলে কোন তফাৎ হয় কিনা?

এই আলোচনায় যাওয়ার আগে আমরা একটু পরের উদ্ধৃতাংশটুকু বিবেচনা করি। "ঢাকার মামা হালিম বিখ্যাত। চল আমরা মামা হালিম খাই। তুমি খাবে এক বাটি, আমি খাব এক বাটি। আমার বাটিটা পরিস্কার নয়, তোমার বাটিটা পরিস্কার।" তো এইখানে বাটি মানে কখন আসলে হালিম আর কখন আসলে সেটা পাত্র? আমরা বুঝতে পারি "তুমি খাবে এক বাটি, আমি খাব এক বাটি" এই কথাগুলোতে বাটি বলতে আসলে সত্যি সত্যি পাত্রটাকে কামড়ে কামড়ে খাওয়ার কথা বলা হচ্ছে না, বরং তুমি এক বাটি পরিমান হালিম খাবে আর আমি এক বাটি পরিমান হালিম খাবো তাই বুঝানো হচ্ছে। এক বাটি হালিম মানে একটা বাটিতে থাকা হালিম। বিষয়গুলোকে চলক আর তার মানের সাথে মিলাও। বাটি ঠিক যেন চলকের মতো আর হালিম হল তার মানের মতো। আবার "আমার বাটিটা পরিস্কার নয়, তোমার বাটিটা পরিস্কার।" এই অংশে বাটি মানে আসলে বাটি নামের পাত্রটা, সেই পাত্রে ঢালা হালিম নয় কোন ভাবেই। তাহলে দেখা যাচ্ছে বাটি বলতে কখনো কখনো আসলে পাত্রটাকে বুঝানো হয় আর কখনো কখনো পাত্রটাতে থাকা হালিমকে বুঝানো হয়। একই ভাবে চলকের নাম উল্লেখ করলে কখনো কখনো চলকটির মানকে বুঝানো হয়, কখনো কখনো আসলে চলকটির জন্য স্মরণিতে (memory) বরাদ্দ জায়গাটুকু বুঝানো হয়।

x = 3; এখানে চলক x বলতে আমরা আসলে চলক x এর জন্য স্মরনিতে (memory) নেয়া জায়গাটুকু বুঝি যেখানে মান 3 কে রাখা হবে। এখানে কোন ভাবেই চলক x এ আগে থেকে বিদ্যমান মানকে বুঝানো হচ্ছে না। খেয়াল করো এখানে চলক x আরোপ = চিহ্নের বাম পাশে আছে। যখন চলক x আসলে স্মরনিতে বরাদ্দকৃত জায়গাকে বুঝায় তখন এটাকে আমরা স্রেফ চলক না বলে আরো স্পষ্ট করে বলবো চলকের বাম-মান (l-value)। তাহলে মনে রেখো চলকের বাম মান দিয়ে আমরা বুঝাবো চলকের জন্য স্মরনিতে নেয়া জায়গাটুকু।

y = x; এখানে চলক y বলতে আমরা চলক y এর জন্য স্মরনিতে বরাদ্দ পাওয়া জায়গাটুকুকে বুঝি। আর চলক y আরোপ = চিহ্নের বামে আছে তাই এখানে চলক y এর বাম-মান ব্যবহৃত হয়েছে। তবে চলক x বলতে এখানে আমরা কেবল তার মানটাকে বুঝি। খেয়াল করো চলক x এর মানটাইতো চলক y এর স্মরনির জায়গাটাতে জমা হবে, চলক x এর জন্য বরাদ্দ জায়গাতো আর গিয়ে চলক y এর জায়গায় লেখা হবে না। আমরা দেখছি এখানে চলক x আরোপ = চিহ্নের ডানে রয়েছে। যখন চলক x আসলে তার মানটাকে বুঝায় তখন আমরা এটাকে বলব চলকের ডান-মান (r-value)। চলকের ডান মান দিয়ে আমরা তাহলে বুঝাবো চলকের যে মান সেটিকে, স্মরণিতে থাকা জায়গাটিকে নয়।

উপরের আলোচনা থেকে আমরা একটা বিষয়ই পরিস্কার করতে চেয়েছি সেটা হলো, আরোপ = চিহ্নের বামে আমরা কেবল এমন কিছু দিতে পারবো যার জন্য স্মরনিতে জায়গা দখল করা আছে,

অর্থাৎ যার বাম-মান (l-value) আছে। আর আরোপ চিহ্নের ডান পাশে আমরা এমন কিছু দিতে পারবো যার মান আছে অর্থাৎ ডান-মান (r-value) আছে। একটা বিষয় খেয়াল করো যার বাম-মান আছে অর্থাৎ স্মরনিতে যার জায়গা আছে তার একটা মানও থাকবেই অর্থাৎ তার ডান-মান থাকবেই, যেমন যে কোন চলকের। কথা হচ্ছে এমন কিছু কি আছে যার ডান মান আছে কিন্তু বাম মান নাই। উত্তর ধরে নিতে পারো আছে। যেমন x = 3; এইখানে 3 এর ডান মান আছে কিন্তু বাম মান নাই। কাজেই কেউ চাইলে 3 = x; লিখতে পারবে না, সংকলন (compile) করার সময় ত্রুটি দেখাবে বলবে "error: lvalue required as left operand of assignment"। একই ভাবে কেউ চাইলে আরোপণ হিসাবে y+3 = x;ও লিখতে পারবে না, একই ত্রুটি (error) দেখাবে, কারণ চলক y এর বাম মান সম্ভব হলেও y + 3 করলে ওইটা আর চলক y থাকে না হয়ে যায় একটা রাশি যার মান হবে y এর মান যোগ 3, কাজেই সেটার কেবল মান থাকে, তার জন্য স্মরনিতে কোন জায়গা থাকে না। বুঝাই যাচ্ছে অন্যদিকে আরোপণ হিসাবে x = y + 3; লিখা যাবে কারণ y + 3 এর ডান-মান আছে অপর দিকে চলক x এর বাম-মান আছে।

৪.৬ আত্ম-শরন আরোপণ (Self-Referential Assignment)

ক্রমলেখ (program) দেখলে আমাদের সাধারণত x = x + 1; বা এই জাতীয় অদ্ভুত কিছু বিষয় নজরে আসে। মুল কথা হলো এই সব ক্ষেত্রে একই চলক (variable) আরোপ (assignment) = চিহ্নের বামেও রয়েছে আবার ডানেও রয়েছে। আমরা সকলে গণিত জানি কম বা বেশী। সেখানে সমীকরণ নিয়ে আমাদের যে ধারনা আছে সেই অনুযায়ী তো x কখনো x + 1 এর সমান হতে পারে না। তাহলে ক্রমলেখতে x = x + 1; এর মতো অর্থহীন বিষয় কেন থাকে?

```
x = x + 1;   // চিহ্ন = গণিতের সমান চিহ্ন নয়, এটি গণনার আরোপণ।
```

আসলে = চিহ্নটি গণিতে আমরা ব্যবহার করি দুটো সংখ্যা তুলনা করে যদি দেখি তারা একে অপরের সমান তাহলে। আমরা তাই ওটাকে গণিতে সমান (equal) চিহ্ন বলে থাকি। কিন্তু গণনার জগতে = চিহ্নটিকে সমান চিহ্ন হিসাবে ব্যবহার না করে বরং আরোপণ (assignment) চিহ্ন হিসাবে ব্যবহার করা হয়। কাজেই কোন ক্রমলেখতে আমরা যখন x = x + 1; দেখি তখন আসলে ওটা কোন ভাবেই গণিতের সমীকরণ নয়, বরং ওইটা গণনার জগতের আরোপণ। সুতরাং গণিতের জগতে ওইটা কোন অর্থ তৈরী না করলেও গণনার জগতে ওটার সুনির্দিষ্ট অর্থ আছে।

আমরা আরোপণ (assignment) নিয়ে আগেই আলোচনা করেছি। ওই আরোপণগুলোর সব-গুলোতে বাম আর ডান উভয় পাশে চলক থাকলেও আলাদা আলাদা চলক ছিল। আর x = x + 1ও আরোপণ তবে এখানে একই চলক আরোপ চিহ্নের বামেও আছে ডানেও আছে। এইরকম আরো-পণ যেখানে একই চলক বামেও আছে ডানেও আছে সেটাকে আমরা বলবো আত্মশরন আরোপণ (self-referential assignment) অর্থাৎ যেখানে একটা চলক নিজের মানের জন্য নিজেরই শরনাপন্ন হয়। আত্মশরণ আরোপণে ডানপাশে চলকটির ডান-মান (r-value) ব্যবহৃত হয়, আর বামপাশে চলকটির বাম-মান (l-value) ব্যবহৃত হয়। এই রকম আরোপণে আসলে কী ঘটে?

```
int x = 3;          // চলক x এ আদি মান আরোপ করা হলো
x = x + 1;          // এখানে আত্ম–শরণ আরোপণ করা হচ্ছে
cout << x << endl;  // চলক x এর মান ফলন দেওয়া হচ্ছে
```

এই রকম আরোপণ বুঝতে গেলে আমরা x = x + 1; বিবৃতিটিকে দুইটি ঘটনায় বিভক্ত করে নিতে পারি। একটা ঘটনা হল ডান পাশে x + 1 হিসাব করা অর্থাৎ x+1 এর মান বা আরো পরিস্কার করে বললে ডান-মান হিসাব করা। আর অন্য ঘটনাটা হল বাম পাশে x এর বাম-মানে অর্থাৎ স্মরনি-তে (memory) x এর জন্য বরাদ্দ করা জায়গায় ডান পাশ থেকে পাওয়া মানটি লিখে দেওয়া। তো

এই দুটো ঘটনার প্রথমটি আগে ঘটবে আর দ্বিতীয়টি পরে ঘটবে। উপরে আমরা x এর আদি মান নিয়েছি 3। এরপর যখন x= x+1; নির্বাহিত (execute) হবে তখন প্রথম ঘটনাটি ঘটবে আগে অর্থাৎ x + 1 মান হিসাব হবে। x এর মান যেহেতু এই অবস্থায় 3 তাই x + 1 হবে 4। মনে করে দেখো এই 4 এর কিন্তু কেবল ডান-মান আছে এর জন্য স্মরনিতে কোন জায়গা দখল করা নেই বা এর কোন বাম-মান নেই। অর্থাৎ এই 4 কোন ভাবেই x চলকের জায়গায় নেই, অন্য কোথাও আছে। যাইহোক এমতাবস্থায় এরপর ঘটবে দ্বিতীয় ঘটনাটি অর্থাৎ এই 4 মানটি গিয়ে লেখা হয়ে যাবে x এর জন্য বরাদ্দ জায়গাতে। আমরা তাই x এর পুরনো মান 3 বদলে সেখানে পাবো এর নতুন মান 4। তাহলে x = x + 1; আত্ম-শরন আরোপণের ফলে চলকের মান এক বেড়ে গেলো।

আত্মশরন আরোপণের আরো নানারকম জটিল অবস্থা আছে যেমন x = x * 3; বা x = x* x + x + 1;। এগুলোর প্রতিটি ক্ষেত্রে আগে ডানপাশের মান হিসাব করা হবে আর তারপর সেই মান বাম পাশে লিখে দেয়া হবে, ফলে চলকটিতে নতুন একট মান থাকবে।

৪.৭ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. সরাসরি ক্রমলেখয়ের (program) ভিতরে দিলে উপাত্ত দিয়ে দিলে সমস্যা কী?

২. উপাত্ত (data) কেনো যোগান (input) নিতে হবে? সুবিধা-অসুবিধা কী কী?

৩. যোগান যাচনা (input prompt) কী? যোগান নেয়ার আগে কেন যাচনা করা উচিৎ?

৪. চলকে (variable) মান আরোপণে (assignment) লক্ষ্য ও উৎসে কী ঘটে বর্ণনা করো।

৫. চলকের বাম-মান আর ডান-মান বলতে কী বুঝো? উদাহরণ দিয়ে ব্যাখ্যা করো।

৬. আরোপণে = চিহ্নের বামে কেন এমন কিছু দেয়া যায় না যার কেবল ডান মান আছে?

৭. আত্ম-শরণ (self-referential) আরোপণ কী উদাহরণ সহ ব্যাখ্যা করো।

৮. দুটি চলকে (variable) থাকা মান বদলাবদলি করবে কেমনে ব্যাখ্যা করো।

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি পূর্ণক (int) ও একটি ভগ্নক (float) যোগান (input) নিয়ে সেগুলো আবার ফলনে (output) দেখায়।

২. এমন একটি ক্রমলেখ (program) রচনা করো যেটি দুটি ভগ্নক (float) সংখ্যা যোগান (input) নিয়ে সংখ্যা দুটি ও তাদের যোগফল ফলনে (output) দেখায়।

৩. এমন একটি ক্রমলেখ (program) রচনা করো যেটি তিনটি পূর্ণক (int) যোগান (input) নিয়ে তাদেরকে যে ক্রমে যোগান নেয়া হয়েছে সেই ক্রমে আবার উল্টোক্রমে দেখাবে। যেমন ভুক্ত সংখ্যা তিনটি যদি হয় পর পর 2 3 1 তাহল সিধা ক্রমে দেখাবে 2 3 1 আবার তাদের উল্টাক্রমে দেখাবে 1 3 2। খেয়াল করো আমরা কিন্তু মানের ক্রম বলছি না।

৪.৭. অনুশীলনী সমস্যা (Exercise Problems)

৪. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একদম ঠিক ঠিক নীচের মতো যোগান (input) ও ফলন (output) উৎপন্ন করে। তুমি কিন্তু পরীক্ষার নম্বরগুলো যোগান নিবে, আর আমরা একেকবার চালানোর সময় এক এক রকম সংখ্যা যোগান দিবো।

```
folafol nirnoyer kromolekho
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _
prothom porikkhai koto? 90
ditiyo porikkhai koto? 75
tritiyo porikkhai koto? 91
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _
shorbo mot number holo 256
```

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

> আমরা ধরে নিচ্ছি যে তুমি দরকারী শির নথি (header) অন্তর্ভুক্ত করা, নামাধার (namespace) std ব্যবহার করা, main বিপাতকের কংকাল লেখা আর সেটার শেষে return EXIT_SUCCESS; লিখে মান ফেরত দেয়া ইত্যমধ্যে ভালো করে শিখে ফেলেছো। তো তুমি যদি নীচে লেখা ক্রমলেখাংশগুলো সংকলন (compile) করে চালাতে (run) চাও, তোমাকে কিন্তু আগে include, namespace, main, return ওইগুলো লিখে নিতে হবে, তারপর main বিপাতকের ভিতরে return এর আগে তুমি আমাদের নীচের অংশগুলো লিখে নিবে। তারপর সংকলন করে ক্রমলেখ চালাবে। আমরা এখন থেকে মোটামুটি এইভাবে ক্রমলেখাংশ দেখাবো।

১. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি পূর্ণক (int) ও একটি ভগ্নক (float) যোগান (input) নিয়ে সেগুলো আবার ফলনে (output) দেখায়।

ফিরিস্তি ৪.৩: যোগান ও ফলনের ক্রমলেখ (Input Output Program)

```
int purnok;
float vognok;

cout << "purnok koto? ";
cin >> purnok;

cout << "vognok koto? ";
cin >> vognok;

cout << "puronok holo " << purnok << endl;
cout << "vognok holo " << vognok << endl;
```

২. এমন একটি ক্রমলেখ (program) রচনা করো যেটি দুটি ভগ্নক (float) সংখ্যা যোগান (input) নিয়ে সংখ্যা দুটি ও তাদের যোগফল ফলনে (output) দেখায়।

ফিরিস্তি ৪.৪: যোগান প্রকিয়ন ফলন (Input Process Output)

```
float prothom, ditiyo;

cout << "songkhya duti koto? ";
cin >> prothom >> ditiyo;

float jogfol = prothom + ditiyo;

cout << "songkhya duti "; //  কোন endl নাই
cout << prothom << " " << ditiyo << endl;

cout << "tader jogfol " << jogfol << endl;
```

৩. এমন একটি ক্রমলেখ (program) রচনা করো যেটি তিনটি পূর্ণক (int) যোগান (input) নিয়ে তাদেরকে যে ক্রমে যোগান নেয়া হয়েছে সেই ক্রমে আবার উল্টোক্রমে দেখাবে। যেমন ভুক্ত সংখ্যা তিনটি যদি হয় পর পর 2 3 1 তাহল সিধা ক্রমে দেখাবে 2 3 1 আবার তাদের উল্টাক্রমে দেখাবে 1 3 2। খেয়াল করো আমরা কিন্তু মানের ক্রম বলছি না।

ফিরিস্তি ৪.৫: যোগানের সিধা ক্রম উল্টা ক্রম (Input Order Reverse Order)

```
int prothom, ditiyo, tritiyo;

cout << "songkhya tinti koto? ";
cin >> prothom >> ditiyo >> tritiyo;

cout << "sidha krome " << prothom << " ";
cout << ditiyo << " " << tritiyo << endl;

cout << "ulta krome " << tritiyo << " ";
cout << ditiyo << " " << prothom << endl;
```

৪. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একদম ঠিক ঠিক নীচের মতো যোগান (input) ও ফলন (output) উৎপন্ন করে। তুমি কিন্তু পরীক্ষার নম্বরগুলো যোগান নিবে, আর আমরা একেকবার চালানোর সময় এক এক রকম সংখ্যা যোগান দিবো।

```
folafol nirnoyer kromolekho
- - - - - - - - - - - - - - -
prothom porikkhai koto? 90
ditiyo porikkhai koto? 75
tritiyo porikkhai koto? 91
- - - - - - - - - - - - - - -
shorbo mot number holo 256
```

ফলাফল প্রক্রিয়াকরণের ক্রমলেখটি আমরা নীচে দেখাচ্ছি।

ফিরিস্তি ৪.৬: ফলাফল প্রক্রিয়ার ক্রমলেখ (Result Processing Program)

```
int prothom, ditiyo, tritiyo;

cout << "folafol nirnoyer kromolekho" << endl;

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "prothom porikkhai koto? ";
cin >> prothom;

cout << "ditiyo porikkhai koto? ";
cin >> ditiyo;

cout << "tritiyo porikkhai koto? ";
cin >> tritiyo;

cout << "- - - - - - - - - - - - - - - -" << endl;

int folafol = prothom + ditiyo + tritiyo;

cout << "shorbo mot number holo ";
cout << folafol << endl;
```

৪.৮ গণনা পরিভাষা (Computing Terminologies)

- সূত্র (formula)
- যাচনা (prompt)
- অদল-বদল (swap)
- বাম-মান (l-value)
- ডান-মান (r-value)
- আত্ম-শরণ (self-reference)

অধ্যায় ৫

গাণিতিক প্রক্রিয়াকরণ (Mathematical Processing)

গাণিতিক প্রক্রিয়াকরণে রাশিতে (expression) গাণিতিক অণুক্রিয়া (operator) ও বিপাতক (function) সমুহ কী ভাবে হিসাব করা হয় আমাদের তা জানতে হবে।

৫.১ একিক অণুক্রিয়া (Unary Operators)

সিপিপিতে একিক (unary) অণুক্রিয়ক ধনাত্নক (positive) + আর ঋণাত্নক (negative) – কী ভাবে কাজ করে ? যথাযথ ক্রমলেখ লিখে উদাহরণ সহ বুঝিয়ে দাও। একিক (unary) অণুক্রিয়ক (operator) একটা উপাদানের (operand) ওপর প্রযুক্ত হয়ে ফলাফল উৎপন্ন করে।

ফিরিস্তি ৫.১: পাটিগণিতের ধনাত্নক ও ঋণাত্নক (Arithmetic Positive Negative)

```
int a = 5;     int const b = -9; // a চলক b ধ্রুবক
cout << "+7 = " << +7 << "     -7 = " << -7 << endl;
cout << "+a = " << +a << "     -a = " << -a << endl;
cout << "+b = " << +b << "     -b = " << -b << endl;
cout << endl;
cout << "+(a*b) = " << +(a*b);     //  a*b হল রাশি
cout << "     -(a*b) = " << (a*b) << endl;
cout << "+abs(b) = " << +abs(b);  //abs() বিপাতক
cout << "     -abs(b) = " << -abs(b) << endl;
```

ফলনাংশ (output segment)

```
+7 = 7     -7 = -7
+a = 5     -a = -5
+b = -9     -b = 9

+(a*b) = -45     -(a*b) = -45
+abs(b) = 9     -abs(b) = -9
```

কোন সংখ্যা, চলক (variable), ধ্রুবক (constant), বিপাতক (function), বা রাশির (expression) সামনে ধনাত্মক চিহ্ন থাকলে তার যে মান সেটিই থাকে, কিন্তু ঋণাত্মক চিহ্ন থাকলে তার চিহ্ন বদলে যায় অর্থাৎ আগে ধনাত্মক থাকলে পরে ঋণাত্মক হয়ে যায় আর আগে ঋণাত্মক থাকলে পরে ধনাত্মক হয়ে যায়। চলক ও ধ্রুবক আগেই জানো। বিপাতক (function) হলো এমন একটা জিনিস যে কিছু যোগান (input) নিয়ে কিছু ফলন (output) দেয়। যেমন cstdlib নামক শির নথিতে (header file) abs(x) নামে একটা বিপাতক আছে যেটি একটি সংখ্যা যোগান নিয়ে তার চিহ্নটুকু বাদ দিয়ে কেবল মানটুকু ফলন হিসাবে ফেরত দেয়। অর্থাৎ abs(3) হলো 3 আবার abs(-3)ও 3। একই ভাবে abs(3.5) হলো 3.5 আবার abs(-3.5)ও 3.5। রাশি (expression) হল সংখ্যা, ধ্রুবক, চলক, অণুক্রিয়ক, বিপাতক মিলে যখন একটা জিনিস তৈরী হয় যার মান হিসাব করা যায় যেমন 3 + x * abs(y) একটি রাশি যেখানে x আর y হল চলক।

৫.২ দুয়িক অণুক্রিয়ক (Binary Operators)

সিপিপিতে দুয়িক (binary) অণুক্রিয়কগুলো যোগ +, বিয়োগ −, গুণ ∗, কী ভাবে কাজ করে? যথাযথ ক্রমলেখ লিখে উদাহরণ সহ বুঝিয়ে দাও। দুয়িক অণুক্রিয়ক (operator) দুটো উপাদানের (operand) ওপর প্রযুক্ত হয়ে ফলাফল উৎপন্ন করে।

ফিরিস্তি ৫.২: পাটিগণিতের যোগ বিয়োগ গুণ (Arithmetic Plus Minus Times)

```
cout << "5 + 3 = " << 5 + 3 << endl;
cout << "5.1 + 3 = " << 5.1 + 3 << endl;
cout << "5.1 + 3.2 = " << 5.1 + 3.2 << endl;
cout << endl;

cout << "5 - 3 = " << 5 - 3 << endl;
cout << "5.1 - 3 = " << 5.1 - 3 << endl;
cout << "5.1 - 3.2 = " << 5.1 - 3.2 << endl;
cout << endl;

cout << "5 * 3 = " << 5 * 3 << endl;
cout << "5.1 * 3 = " << 5.1 * 3 << endl;
cout << "5.1 * 3.2 = " << 5.1 * 3.2 << endl;
cout << endl;
```

উপরের ক্রমলেখাংশ (program segment) খেয়াল করো। আর তার সাথে নীচের ফলনাংশ (output segment) মিলিয়ে নাও। লক্ষ্য করো আমরা তিনটি করে যোগ, বিয়োগ, আর গুণ করেছি। যোগ, বিয়োগ, বা গুণ আমরা ভালোই জানি, নতুন করে শেখার কিছু নাই। তবে একটি বিষয় খেয়াল করতে হবে। সেটি হলো উপাত্তের ধরণ কেমন? আর এ কারণেই আমরা প্রতিটি অণুক্রিয়ার (operator) জন্যে তিনটি করে উদাহরণ নিয়েছি। প্রতিটি অণুক্রিয়ার উদাহরণগুলোর প্রথম সারিতে খেয়াল করো। সেখানে উপাদান (operand) হিসাবে আমরা দুটো পূর্ণকের যোগ, বিয়োগ বা গুণ করেছি, যেমন 5 + 3, 5 − 3 আর 5 ∗ 3। ফলাফল হিসাবে যা পেয়েছি তাও একটি পূর্ণক, যেমন 8, 2, আর 15। এবার প্রতিটি অণুক্রিয়ার জন্য তৃতীয় সারিতে খেয়াল করো। সেখানে উপাদান (operand) হিসাবে আমরা দুটো ভগ্নক যোগ, বিয়োগ বা গুণ করেছি, যেমন 5.1 + 3.2, 5.1 − 3.2 আর 5.1 ∗ 3.2। ফলাফল হিসাবেও আমরা পেয়েছি একটি ভগ্নক যেমন

8.3, 1.9, আর 16.32। তারপর প্রতিটি অণুক্রিয়ার জন্য দ্বিতীয় সারিতে খেয়াল করো। উপাদান হিসাবে একটি ভগ্নক ও একটি পূর্ণক যোগ, বিয়োগ বা গুণ করা হয়েছে যেমন 5.1 + 3, 5.1 − 3, আর 5.1 * 3। আর ফলাফল এসেছে একটি ভগ্নক যেমন 8.1, 2.1, আর 15.3, যেগুলোর কোনটিই পূর্ণক নয়। উপাদানদুটো একটা ভগ্নক হওয়ায় ফলাফলও ভগ্নক হয়ে গেছে।

ফলনাংশ (output segment)

```
5 + 3 = 8
5.1 + 3 = 8.1
5.1 + 3.2 = 8.3

5 − 3 = 2
5.1 − 3 = 2.1
5.1 − 3.2 = 1.9

5 * 3 = 15
5.1 * 3 = 15.3
5.1 * 3.2 = 16.32
```

তাহলে উপরের আলোচনা থেকে আমরা দেখলাম কোন অণুক্রিয়ার (operator) যদি দুটি উপাদানই (operand) একরকম হয় তাহলে ফলাফলও সেই রকমই হয়। যেমন উপাদান দুটোই int হলে ফলাফলও int; উপাদান দুটোই float হলে ফলাফলও float। আর যদি দুটো উপাদান দুরকম হয় যেমন একটি পূর্ণক বা int আর একটি ভগ্নক বা float তাহলে ফলাফল হবে ভগ্নক বা float। গণিতে আমরা জানি পূর্ণক সংখ্যাগুলো একই সাথে ভগ্নকও যেমন 3 আসলে 3.0, কিন্তু একটি ভগ্নক কিন্তু পূর্ণক নাও হতে পারে যেমন 5.1 ভগ্নক কিন্তু একে পূর্ণক হিসাবে লেখা সম্ভব নয়। আর এ কারণে কোন অণুক্রিয়া (operator) প্রয়োগের পূর্বে উপাদান (operand) দুটো দুরকম হলে প্রথমে পূর্ণকটিকে ভিতরে ভিতরে ভগ্নকে রূপান্তর করে নেয়া হয়, আর তারপর যোগ, বিয়োগ বা গুণ করা হয় দুটোকে ভগ্নক হিসাবে নিয়েই। এই যে ভিতরে ভিতরে পূর্ণকটি ভগ্নকে রূপান্তর করা হয় এটা এক রকমের উপাত্ত প্রকারান্তর (type casting)। উপাত্ত প্রকারান্তর নিয়ে আমরা পরে আরো বিস্তারিত জানবো, আপাতত পূর্ণক থেকে ভগ্নকে প্রকারান্তর মনে রাখো।

৫.৩ ভাগফল ও ভাগশেষ (Division and Remainder)

সিপিপিতে দুয়িক (binary) অণুক্রিয়ক ভাগফল (division) ও ভাগশেষ (remainder) কী ভাবে কাজ করে? যথাযথ ক্রমলেখ লিখে উদাহরণ সহ বুঝিয়ে দাও। তুমি ইত্যমধ্যে জেনেছো দুয়িক অণুক্রিয়ক (operator) দুটো উপাদানের (operand) ওপর প্রযুক্ত হয়ে ফলাফল উৎপন্ন করে।

ফিরিস্তি ৫.৩: পাটিগণিতের ভাগফল অণুক্রিয়া (Arithmetic Division Operation)

```
cout << "13 / 5 = " << 13 / 5 << endl;
cout << "13.0 / 5 = " << 13.0 / 5 << endl;
cout << "13 / 5.0 = " << 13 / 5.0 << endl;
cout << "13.0 / 5.0 = " << 13.0 / 5.0 << endl;
```

ভাগফলের উপাত্ত প্রকরণ (data type) কেমন হবে সেই নিয়ম আসলে যোগ, বিয়োগ, বা গুণের মতো একই। যদি দুটো উপাদানই (operand) এক রকমের হয় তাহলে ফলাফলও হবে

সেই রকমেরই। কিন্তু উপাদান দুটোর একটি যদি হয় পূর্ণক বা int আরেকটি ভগ্নক বা float তাহলে ফলাফল হবে একটি ভগ্নক বা float। এখানেও ভিতরে ভিতরে int প্রথমে float এ প্রকারান্তর (type casting) হয়ে যাবে, ভাগের কাজটি হবে উপাত্ত প্রকারান্তর হবার পরে। উপাত্ত প্রকারান্তর ছাড়াও ভাগের ক্ষেত্রে ভাগশেষ থাকবে কি থাকবে না সেটার একটা ব্যাপার আছে।

ফলনাংশ (output segment)

```
13 / 5 = 2
13.0 / 5 = 2.6
13 / 5.0 = 2.6
13.0 / 5.0 = 2.6
```

উপরের ফলনাংশ লক্ষ্য করো, যদি ভাগের উপাদান দুটোর যেকোন একটিও ভগ্নক হয়, যেমন শেষের তিন সারি, তাহলে কিন্তু ভাগশেষের কোন ব্যাপার থাকে না, ফলে আমরা সেক্ষেত্রে ভাগফল পাই 2.6। কিন্তু ভাগের ক্ষেত্রে যদি দুটো উপাদানই পূর্ণক হয়, যেমন প্রথম সারি তাহলে ভাগটি কিন্তু একটু আলাদা। যেমন 13 / 5 করলে আমরা ফলাফল পাই 2 কারণ আমরা জানি এক্ষেত্রে 3 অবশিষ্ট থাকে। ভাগের ক্ষেত্রে আরো একটি গুরুত্বপূর্ণ বিষয় আছে তা হলো উপাদানের পূর্ণকগুলো ধনাত্মক না ঋণাত্মক। কারণ ঋণাত্মক সংখ্যার ভাগ একটু বিটকেলে হতে পারে। সব মিলিয়ে পূর্ণ সংখ্যার ভাগ আরো বিস্তারিত করে আমরা ভাগশেষের সাথে মিলিয়ে নীচে আলোচনা করবো। তবে একটা কথা মনে রাখবে ভাগের ক্ষেত্রে যদি ভাজক শুন্য হয় যেমন 13 / 0 তাহলে তোমার ক্রমলেখ চালানোর (run) সময় divide by zero বা শুন্য দিয়ে ভাগ নামে ত্রুটিবার্তা (error message) দেখিয়ে বন্ধ হয়ে যাবে। এই রকম ত্রুটি সংকলনের (compile) সময় ধরা পড়ে না, কেবল চালানোর (run) সময় বা নির্বাহ (execute) করার সময় ধরা পড়ে, তাই এদেরকে বলা হয় চলা-কালীন (run-time) বা নির্বাহ-কালীন (execution-time) ত্রুটি।

ফিরিস্তি ৫.৪: পাটিগণিতের ভাগশেষ অণুক্রিয়া (Arithmetic Remainder Operation)

```
cout << "13 / 5 = " << 13 / 5 << "    ";
cout << "13 % 5 = " << 13 % 5 << endl;

cout << "13 / -5 = " << 13 / -5 << "    ";
cout << "13 % -5 = " << 13 % -5 << endl;

cout << "-13 / 5 = " << -13 / 5 << "    ";
cout << "-13 % 5 = " << -13 % 5 << endl;

cout << "-13 / -5 = " << -13 / -5 << "    ";
cout << "-13 % -5 = " << -13 % -5 << endl;

// নীচের সারিগুলো সংকলন (compile) হবে না, ভগ্নকে ভাগশেষ হয় না
// cout << "13.0 % 5 = " << 13.0 % 5 << endl;
// cout << "13.0 % 5.0 = " << 13.0 % 5.0 << endl;
// cout << "13.0 / 5.0 = " << 13.0 / 5.0 << endl;
```

যাইহোক সবশেষে আমরা ভাগশেষ দেখি। ভাগের ক্ষেত্রে আমরা আলোচনা করেছি ভগ্নক বা float এর জন্য ভাগশেষের কোন ব্যাপার নেই। কাজেই ভাগশেষ অণুক্রিয়ার (operator) উপাদান (operand) দুটোর যে কোন একটিও যদি ভগ্নক হয়, তাহলে ভাগশেষ মোটামুটি অর্থহীন

হয়ে যায়। কাজেই এমন কিছু আমাদের ক্রমলেখতে (program) লিখলে সংকলন (compile) করার সময় ত্রুটি (error) আসবে। নীচের ক্রমলেখাংশের শেষের তিনটি সারি দেখতে পারো যে-গুলো টীকা হিসাবে রাখা আছে। ওইগুলো টীকা না করে সামনের // হেলানো দাগ দুটো তুলে দিলে ক্রমলেখয়ের অংশ হয়ে যাবে, আর তখন সংকলন করলে ত্রুটি আসবে, করে দেখতে পারো।

একটা বিষয় খেয়াল করেছো, এখানে আমরা কিন্তু টীকার (comment) হেলানো // চিহ্ন দুটোর একরকমের অপব্যবহার করেছি। উপরের ক্রমলেখাংশের শেষ তিনটি সারি আসলে কোন ভাবেই প্রকৃত টীকা নয়। ওগুলোতো বাংলায় বা ইংরেজীতে লেখা নয়, ওগুলো সিপিপিতে লেখা আর টীকা চিহ্ন তুলে নিলেই ওগুলো ক্রমলেখয়ের অংশ হয়ে যাবে সহজেই। তবু কেন এখানে আম-রা ওগুলোকে টীকার ভিতরে রাখলাম? এটা আসলে একটা খুবই উপকারী কৌশল। টীকার ভিতরে রাখলে যেহেতু সেটা ক্রমলেখয়ের ঠিক অংশ থাকে না, সংকলন হয় না, কোন ত্রুটি আসার ব্যাপার নাই, আমরা তাই মাঝে মাঝে কিছু কিছু সিপিপিতে লেখা অংশও টীকার ভিতরে রাখি। ক্রমলেখ (program) লেখার সময় আমরা নানান কিছু পরীক্ষা নিরীক্ষা করি, এভাবে করি, ওভাবে করি। তখন যে অংশগুলো ওই সময় দরকার নাই, চাইলে সেগুলো তো মুছে ফেলা যায়, কিন্তু মুছে ফেল-লেই তো তোমাকে পরে আবার কষ্ট করে লিখতে হতে পারে। এমতাবস্থায় তুমি যদি ওই অদরকারী অংশটুকুতে টীকা দিয়ে (commenting) দাও, ব্যস হয়ে গেলো। কোন ঝামেলা নাই, পরে ওই অংশটুকু আবার দরকার হলেই টীকা তুলে (uncomment) নিবে। কী চমৎকার কৌশল তাই না! আমরা সবাই এটি হরদম ব্যবহার করি। এখন থেকে এই কৌশল কাজে লাগাবে, কেমন!

ফলনাংশ (output segment)

```
13 / 5 = 2      13 % 5 = 3
13 / -5 = -2      13 % -5 = 3
-13 / 5 = -2      -13 % 5 = -3
-13 / -5 = 2      -13 % -5 = -3
```

এবারে ভাগশেষের ফলাফলের দিকে নজর দেই। ভাগফল সহ আলোচনার সুবিধার জন্য উপ-রের ক্রমলেখাংশ (program segment) আর ফলনাংশে (output segment) আমরা ভা-গশেষের সাথে সাথে ভাগফলও দেখিয়েছি। আমরা আগেই আলোচনা করেছি ভাগশেষ করা যায় কেবল পূর্ণকের জন্য। ভাগ করলে যা অবশেষ থাকে তাই ভাগশেষ। কিন্তু পূর্ণক তো ধনাত্মকও (positive) হতে পারে, ঋণাত্মকও (negative) হতে পারে। আসলে ঋণাত্মক সংখ্যার ভাগশেষ নিয়েই যতো জটিলতা সৃষ্টি হয়। ঋণাত্মক সংখ্যার ভাগশেষ নিয়ে নানান রকম নিয়ম আছে, আমরা এখানে আলোচনা করছি ideone.com এ যে নিয়মে ভাগশেষ হয়, সেটা নিয়ে। তুমি যে সংকলক (compiler) দিয়ে ক্রমলেখ সংকলন (compile) করবে, জেনে নিও সেখানে কেমন হয়। কারো কাছে থেকে জেনে নিতে পারো। অথবা নিজেই উপরের ক্রমলেখাংশ (program segment) এর মতো করে ক্রমলেখ তৈরী করে চালিয়ে দেখে নিতে পারো। তেমন কঠিন কিছু নয়।

যাইহোক উপরের ফলনাংশ খেয়াল করো। সেখানে কিন্তু কোন ভগ্নক নেই, সবগুলোই পূর্ণক, তবে ধনাত্মক ও ঋণাত্মক আছে। খেয়াল করো ভাগফল ও ভাগশেষ উভয় ক্ষেত্রে কেবল মানটা পাওয়া যায় চিহ্ন বিবেচনা না করলে। যেমন চারটা ব্যাপারের সবগুলোতেই চিহ্ন বাদ দিলে ভাজক (divisor) আর ভাজ্য (dividend) হয় কেবল 5 আর 13। 13 কে 5 দিয়ে ভাগ করলে ভাগফল হয় 2 আর ভাগশেষ হয় 3। এই পর্যন্ত সবগুলো ব্যাপারেই ঠিক আছে, কিন্তু গোলমাল বাঁধে কেবল চিহ্ন নিয়ে, ভাগফল বা ভাগশেষ কখন ধনাত্মক + হবে আর কখন ঋণাত্মক - হবে। ভাগফলের ক্ষেত্রে খেয়াল করো যখনই সংখ্যা দুটোর চিহ্ন একই রকম তখন ভাগফল ধনাত্মক যেমন প্রথম ও চতুর্থ সারি, আর যখনই তারা বিপরীত চিহ্নের তখনই ভাগফল ঋণাত্মক যেমন দ্বিতীয় ও তৃতীয় সারি। ভাগশেষের ক্ষেত্রে চিহ্ন নির্ভর করে ভাজ্য (dividend) এর ওপর, ভাজকের ওপর নয়। ভাজ্য যখনই ধনাত্মক যেমন 13, ভাগশেষ তখন ধনাত্মক + হয়েছে। আর ভাজ্য যখন ঋণাত্মক যেমন -

13 তখন ভাগশেষ ঋণাত্মক - হয়েছে। ভাগশেষের চিহ্ন 5 বা -5 এর চিহ্নের ওপর নির্ভর করে নাই। একটা গুরুত্বপূর্ণ বিষয় আগেই বলেছি, ভাগফল ও ভাগশেষের ক্ষেত্রে ভাজক যদি শুন্য হয় তাহলে তোমার ক্রমলেখক চালানোর সময় divide by zero বা শুন্য দিয়ে ভাগ নামে ত্রুটিবার্তা দেখিয়ে বন্ধ হয়ে যাবে। এই রকম ত্রুটি সংকলনের (compile) সময় ধরা পড়ে না, কেবল চালানোর (run) সময় ধরা পড়ে, তাই এদেরকে বলা হয় চলা-কালীন ত্রুটি (run-time error)।

উপরের উদাহরণগুলোতে আমরা যদিও কেবল সংখ্যাই সরাসরি ব্যবহার করেছি, তুমি কিন্তু চাইলে কোন চলক (variable) বা ধ্রুবক (constant) ব্যবহার করতে পারতে। তুমি চাইলে কোন রাশি (expression) বা বিপাতক (function) ও ব্যবহার করতে পারতে। আসলে ডান-মান (rvalue) আছে এরকম যে কোন কিছুই এখানে ব্যবহার করা যেতে পারে। এই আলোচনাগুলো একিক অণুক্রিয়ার সময়ই আলোচনা করা হয়েছে, তবুও আবার বলি। বিপাতক (function) এমন একটা জিনিস যে কিছু যোগান (input) নিয়ে কিছু ফলন (output) দেয়। যেমন cstdlib নামক শির নথিতে (header file) abs(x) নামে একটা বিপাতক আছে যেটি একটি সংখ্যা যোগান নিয়ে তার চিহ্নটুকু বাদ দিয়ে কেবল মানটুকু ফলন হিসাবে ফেরত দেয়। অর্থাৎ abs(3) হলো 3 আবার abs(-3)ও 3। একই ভাবে abs(3.5) হলো 3.5 আবার abs(-3.5)ও 3.5। রাশি (expression) হল সংখ্যা, ধ্রুবক, চলক, অণুক্রিয়ক, বিপাতক মিলে যখন একটা কিছু তৈরী করা হয় যার মান আছে সেটি, যেমন 3 + x * abs(y) একটি রাশি যেখানে x আর y হল চলক।

```
int a = 4, b = -3;
int const c = 5;

a + 3, c / b, b * c;      //চলক, ধ্রুবক, সংখ্যা
a = c % abs(b);           // abs(b) হল বিপাতক
a = a - ( b * c);         // b * c হল রাশি
```

৫.৪ আরোপণ অণুক্রিয়া (Assignment Operator)

আরোপণে (assignment) চলকের জন্য স্মরণিতে (memory) বরাদ্দকৃত স্থানে মান ভরে দেয়ার ব্যাপারটা আমরা আগে দেখেছি। কিন্তু আরোপণ আসলে একটা অণুক্রিয়াও (operator) বটে। আরোপণ একটা অণুক্রিয়া এই কথার মানে কী? আমরা আরোপণ নিয়া কী কী করতে পারবো?

আরোপণ (assignment) একটা অণুক্রিয়া (operator) এই কথার মানে হলো আরোপণ কিছু উপাদানের (operand) ওপর প্রযুক্ত হয়ে একটি ফলাফল উৎপন্ন করে। সত্যি বলতে গেলে যোগ, বিয়োগ, গুণ বা ভাগের মতো আরোপণও আসলে একটা দুয়িক (binary) অণুক্রিয়া। কাজেই এটি দুটি উপাদানের (operand) ওপর প্রযুক্ত হয়। আরোপণের বাম পাশে একটা উপাদান থাকে যার বাম-মান থাকতে হবে অর্থাৎ যার জন্য স্মরণিতে (memory) জায়গা বরাদ্দ থাকতে হবে, যেমন চলক। আর আরোপণের ডানে থাকতে হবে এমন কিছু যার ডান-মান বা মান আছে, যেমন চলক (variable), ধ্রুবক (constant), বিপাতক (function) বা রাশি (expression)। কথা হচ্ছে আরোপণের ফলে উৎপন্ন হওয়া ফলাফলটা কী? আসলে যে মানটি আরোপণের বামপাশের চলকে আরোপিত হয় সেই মানটিই আরোপণ অণুক্রিয়ার ফলাফল হিসাবেও বিবেচনা করা হয়।

```
int v = 3, w = -5, x, y, z; //  ভগ্নকও নেয়া যেতে পারে
x = v + 5;     //  চলক x এর মান 8, আরোপণের ফলাফলও 8
y = abs(w);    //  চলক y এর মান 5, আরোপণের ফলাফলও 5
z = x + y;     //  চলক z এর মান 13, আরোপণের ফলাফলও 13
```

উপরে ক্রমলেখাংশে v + 5 বা 3 + 5 অর্থাৎ 8 আরোপিত হয়েছে x এ। তারপর, abs(w) বিপাতক w বা -5 এর মান হতে চিহ্ন ছাড়া 5 ফেরত দিয়েছে যা আরোপিত হয়েছে y চলকে। আর শেষে x + y বা 8 + 5 অর্থাৎ 13 আরোপিত হয়েছে z চলকে।

তাহলে অন্যান্য অণুক্রিয়ার মতো আরোপণ অণুক্রিয়ারও যেহেতু একটি ফলাফল আছে কাজেই সেই ফলাফলটি অন্য কোন চলক যার বাম-মান আছে তাতে আবারও আরোপন করা সম্ভব!

```
int v = 3, w = -5, x, y, z; //  ভগ্নকও নেয়া যেতে পারে
x = (v + w);// যোগ অণুক্রিয়ার ফলাফল একটি চলকে আরোপণ
z = (y = x);// ডানের আরোপণের ফলাফল বামেরটিতে আরোপণ
z = v * w;  // গুণ আগে হবে, গুণফল আরোপণ তারপরে হবে
z = y = x;  // ডানের আরোপন আগে, সেই ফল নিয়ে বামের আরোপন
```

সুতরাং কেউ যেমন অনেকগুলো যোগ পরপর লিখতে পারে x + y + z + 3, ঠিক তেমনি চাইলেই কেউ অনেকগুলো আরোপণও পরপর লিখতে পারে যেমন z = y = x = w। তবে কোন বন্ধনী নাই ধরে নিলে, যোগের ক্ষেত্রে সাধারণত সবচেয়ে বামের যোগটি থেকে শুরু হয়ে যোগগুলো পরপর বাম থেকে ডানে একে একে হতে থাকে। আর আরোপণের (assignment) এর ক্ষেত্রে সবচেয়ে ডানে আরোপণ হতে শুরু করে আরোপণগুলো ডান থেকে বামে একে একে হতে থাকে।

```
int x = 1, y = 2; z = 3;  // আদি মান আরোপণ

x + (y = 3);     // y হলো 3, ফলাফল 1 + 3 বা 4
y = x + (z = 4);  // z হলো 4, y হলো 1 + 4 বা 5
z = 5 + (y = z -3); // y হলো 4 - 3 বা 1, z হলো 5 + 1
```

উপরের উদাহরণের শেষ তিনটি সারি খেয়াল করো। চলক ঘোষনার পরের সারির বিবৃতিতে (statement) x + (y = 3); প্রথমে বন্ধনীর ভিতরে y এর মান 3 আরোপণ (assign) হবে আর আরোপণের (assignment) ফলাফলও হবে 3, যা x এর মান 1 সাথে যোগ হয়ে যোগফল হবে 4। এই 4 হলো পুরো রাশিটির মান। এরপরের বিবৃতিতে y = x + (z = 4); প্রথমে বন্ধনীর ভিতরে z এর মান আরোপ হবে 4 আর ফলাফল ও 4, আর তারপর 4 ও x এর মান 1 এর সাথে যোগ হয়ে হবে 5 যা গিয়ে y চলকে আরোপিত হবে। এবারে আসি শেষ বিবৃতিতে z = 5 + (y = z -3); প্রথমে বন্ধনীর ভিতরে z - 3 হিসাব হবে, z এর মান ঠিক আগের সারিতে হয়েছে 4 সাথে 3 বিয়োগ হলে হয় 1 যা y এ আরোপিত হবে আর আরোপণের ফলাফলও (result) হবে 1। এরপর সেই 1 আর 5 যোগ হয়ে ফল হবে 6 যা z এর ভিতরে আরোপিত হবে।

৫.৫ যৌগিক আরোপন (Compound Assignment)

যৌগিক আরোপণ (compound assignment) কী? সিপিপিতে যৌগিক আরোপণ কী ভাবে আরোপণের সাথে অন্য একটি অণুক্রিয়ার (operator) যোজন (composition) ঘটায়? আত্ম-শরণ (self referential) আরোপণের সাথে যৌগিক আরোপণের সম্পর্ক কী?

যৌগিক আরোপন হলো আরোপনের সাথে আর একটি অণুক্রিয়ার যোজন (composition)। আরোপন = এর সাথে যোগ + এর যোজন ঘটানোর ফলে নতুন যে অণুক্রিয়ক তৈরী হয় সেটি যোগ-আরোপণ +=। একই ভাবে আরোপন = ও বিয়োগ - যুক্ত হয়ে তৈরী হয় বিয়োগ-আরোপণ -=, তারপর একই ভাবে গুণ আরোপণ *=, ভাগফল আরোপণ /= আর ভাগশেষ আরোপণ %=।

```
x += 13;        // এর মানে আসলে x = x + 13;
x -= 7;         //  এর মানে আসলে x = x - 7;
y *= x;         // এর মানে আসলে y = y * x;
z /= x + y;     // এর মানে আসলে z = z / (x + y);
z %= abs(3);    // এর মানে আসলে z = z % abs(3);
```

তাহলে উপরের উদাহরণগুলো থেকে দেখা যাচ্ছে প্রতিটি যৌগিক আরোপণ আসলে এক এক-টি আত্ন-শরণ আরোপন (self-referential assignment)। যৌগিক আরোপণের বাম পাশে যে চলকটি থাকে সেটির মানের সাথে সংশ্লিষ্ট পাটিগণিতীয় অণুক্রিয়া যেমন যোগ, বিয়োগ, গুণ, ভাগফল, বা ভাগশেষ হিসাব করা হয়, আর তারপর ফলাফলটি ওই চলকটিতেই আরোপ করা হয়। আসলে যৌগিক আরোপণগুলো ক্রমলেখ রচনার সময় কষ্ট স্রেফ কিঞ্চিত কমানোর জন্য তৈরী করা হয়েছে। অনেক সময় আরোপনের বাম পাশে যেটি থাকবে সেটি সহজ সরল চলক না হয়ে অন্য কিছু হতে পারে যেটি হয়তো খুবই বড়, সেটির অবশ্যই বাম-মান (l-value) আছে অর্থাৎ তার জন্য স্মর-ণিতে (memory) জায়গা দখল করা আছে। যেমন ধরো নীচের উদাহরণে আমরা সাজন (array) ব্যবহার করছি, শ্রেণী (class) ব্যবহার করছি, এগুলো কী এখনই তা জানতে চেয়ো না, আমরা পরে বিস্তারিত করে শিখবো ওগুলো। খালি খেয়াল করো প্রথম দু সারিতে কী ভাবে লম্বা একটা জিনিস আরোপ = চিহ্নের বাম ও ডান উভয় পাশেই আছে। আর খেয়াল করো শেষের সারির বিবৃতিটি: যৌগিক আরোপণ ব্যবহার করে ওই একই বিষয় কত চমৎকার করে সংক্ষেপে লেখা গেছে।

```
this->amarSajonCholok[suchok] =
    this->amarSajonCholok[suchok] + amarbriddhi;

this->amarSajonCholok[suchok] += amarbriddhi;
```

তাহলে দেখলে তো একই জিনিস আরোপ = চিহ্নের বাম পাশে একবার আবার পরক্ষণেই আরোপ = চিহ্নের ডানপাশেও একবার লিখতে হবে, এটি বেশ বিরক্তিকর, আর দেখতেও কত বিরক্তিকর লাগে। তারচেয়ে যৌগিক আরোপণ সংক্ষিপ্ত আর বুঝাটাও সহজ। ফলাফলের হিসাবে উভয় ক্ষেত্রে কিন্তু আমরা একই ফলাফল পাবো। তবে মনে রেখো ক্রমলেখ (program) চালাতে সময় কম লাগবে নাকি বেশী লাগবে সেইক্ষেত্রে কিন্তু যৌগিক আরোপণের কোন ভুমিকা নেই।

৫.৬ হ্রাস ও বৃদ্ধি অণুক্রিয়া (Increment and Decrement)

সিপিপিতে লেখা ক্রমলেখতে (program) আমরা ++ বা -- প্রায়ই দেখতে পাই। এইগুলো কী? একটা যোগ বা বিয়োগ চিহ্ন দেখেছি কিন্তু দুটো যোগ বা বিয়োগ একসাথে তো আজব ব্যাপার! দুটো যোগ বা বিয়োগ এক সাথে দেয়ার সুবিধা-অসুবিধা কী? ক্রমলেখ কি এতে দ্রুত চলে?

```
int x = 6, y; //  দুটো চলক একটার আদিমান আছে, আরেকটার নাই
++x;        // এক বেড়ে x হলো 7, y জানিনা কারণ আদিমান নেই
x++;        // এক বেড়ে x হলো 8, y জানিনা কারণ আদিমান নেই
y = ++x;    // এক বেড়ে x হলো 9, তারপর y এ 9 আরোপিত হলো
y = x++;    // প্রথমে y হলো x এর সমান বা 9, পরে x হলো 10
```

উপরের ক্রমলেখাংশ (program segment) খেয়াল করো। দুটো চলক (variable) নেয়া হয়েছে x আর y। চলক x এর আদিমান (initial value) দেয়া হয়েছে 6, কিন্তু y এর আদি মান দে-

য়া হয় নি। এরপর দ্বিতীয় আর তৃতীয় বিবৃতিতে রয়েছে ++x; আর x++;, খেয়াল করো উভয় ক্ষেত্রে x এর মান এক করে বেড়েছে, এ কারণে অবশ্য ++ কে বলা হয় বৃদ্ধি অণুক্রিয়ক (increment operator)। বৃদ্ধি অণুক্রিয়ক ++ চলকের আগেই দেয়া হউক আর পরেই দেয়া হউক ফলাফল কিন্তু একই। অবশ্য বৃদ্ধি ++ আগে ব্যবহার করলে এটিকে পূর্ব-বৃদ্ধি (pre-increment) আর পরে ব্যবহার করলে এটিকে উত্তর-বৃদ্ধি (post-increment) বলা হয়।

তবে বলে রাখি বৃদ্ধি অণুক্রিয়কের (increment operator) সাথে কিন্তু এমন কিছু ব্যবহার করতে হবে যার বাম-মান (l-value) রয়েছে অর্থাৎ স্মরণিতে (memory) জায়গা দখল করা আছে। চলকের (variable) যেহেতু বাম-মান আছে তাই আমরা চলক x ব্যবহার করতে পারলাম। কিন্তু তুমি যদি চাও ++3 বা 3++ লিখবে যাতে 4 পাওয়া যায় অথবা লিখবে (x+3)++ বা ++(x+3), তা লিখতে পারবে না, সংকলন (compile) ত্রুটি হবে। ত্রুটি হওয়ার কারণ 3 সংখ্যা (number) বা x+3 রাশির (expression) ডান-মান (r-value) তথা মান (value) আছে কিন্তু তাদের বাম-মান (l-value) তথা স্মরণিতে (memory) জায়গা দখল করা নেই। দরকার নেই তবুও বলে রাখি, তুমি কিন্তু ++ এর সাথে চলক x এর বদলে ধ্রুবক (constant) জাতীয় কিছু তো এমনিতেই ব্যবহার করতে পারবে না, কারণ ধ্রুবকের তো মান বদলানো যায় না।

যাইহোক ++ আগেই দেই আর পরেই দেই ++x বা x++ আসলে x+=1; অর্থাৎ x = x+1; এর সমতুল এবং সংক্ষিপ্ত রূপ বলতে পারো। লক্ষ্য করো বৃদ্ধিতে ++ যে 1 বৃদ্ধি ঘটে সেই ব্যাপারটা কিন্তু উহ্য থাকে। ফলে ++ কেবল একটা উপাদানের (operand) ওপর প্রযুক্ত হয় বলে মনে হয়। আর তাই ++ কে একটি একিক (unary) অণুক্রিয়ক (operator) বলা হয়। কথা হচ্ছে এই একিক অণুক্রিয়ার ফলাফলটা কী? ফলাফল তো আমরা আগেই দেখেছি, মান এক বেড়ে যাওয়া। সেটা ঠিক, কিন্তু তাছাড়াও বৃদ্ধি অণুক্রিয়ার (increment operator) ফলাফলে কিছু গুরুত্বপূর্ণ বিষয় আছে যে কারণে পূর্ব-বৃদ্ধি (pre-increment) আর উত্তর-বৃদ্ধি (post-increment) আলাদা।

পূর্ব-বৃদ্ধি (pre-increment) আর উত্তর বৃদ্ধি (post-increment) যে আলাদা তা পরিস্কার হবে উপরের ক্রমলেখাংশের (program segment) শেষের সারি দুটো দেখলে। যখন y = ++x; করা হয়েছে তখন x এর মান আগে বেড়ে হয়েছে 9 আর তারপর xএর সেই বেড়ে যাওয়া মান 9ই y এ আরোপিত (assign) হয়েছে। কিন্তু যখন y = x++; তখন কিন্তু খেয়াল করো আগে x এর মান y এ আরোপিত হয়েছে ফলে y হয়েছে 9 আর তারপর x এর মান বেড়েছে 1 ফলে হয়েছে 10। আচ্ছা y = ++x; আর y = x++; এ দুটোকে যদি আমরা বৃদ্ধি ++ ব্যবহার না করে লিখতাম তাহলে কেমন হতো? আমাদের অবশ্যই দুটো করে বিবৃতি লিখতে হতো। নীচে লক্ষ্য করো y = ++x; এ আগে মান বাড়ানো পরে আরোপণ, আর y = x++; এ আগে আরোপণ পরে মান বাড়ানো। আশা করা যায় পূর্ব-বৃদ্ধি (pre-) ও উত্তর-বৃদ্ধির (post-increment) তফাৎ পরিস্কার হয়েছে।

```
x = x + 1;   //  y= ++x; এ x এর মান বৃদ্ধি আগে ঘটবে
y = x;       //  y= ++x; এ y তে x এর মান আরোপন পরে

y = x;       //  y= x++; এ y তে x এর মান আরোপন আগে
x = x + 1;   //  y= x++; এ x এর মান বৃদ্ধি তার পরে
```

পূর্ব-বৃদ্ধি (pre-increment) আর উত্তর-বৃদ্ধির (post-increment) আরো একটা পার্থক্যও জানা দরকার অবশ্য। সেটা হলো পূর্ব-বৃদ্ধির ফলাফল আসলে একটা বাম-মান (l-value) এক্ষেত্রে চলকটির বাম-মান, অন্যদিকে উত্তর-বৃদ্ধির ফলাফল আসলে একটা ডান-মান (r-value)। আগেই বলেছি বৃদ্ধি অণুক্রিয়ার সাথে ব্যবহৃত উপাদানের (operand) অবশ্যই বাম-মান থাকতে হবে। ফলে উত্তর-বৃদ্ধির ফলাফলের ওপরে আবার কোন বৃদ্ধিই চালানো যায় না, কিন্তু পূর্ব-বৃদ্ধির ফলাফলের ওপর চালানো যায়। তুমি যদি পরীক্ষা করতে চাও তাহলে ++++x; বা (++x)++; চে-

ষ্টা করো, সংকলন (compile) হয়ে যাবে, কিন্তু x++++ বা ++(x++) চেষ্টা করো, সংকলন হবে না, ত্রুটি (error) আসবে পরের বৃদ্ধিটার জন্য "l-value required"। তুমি যদি স্রেফ ++x++; লিখো, এটা কিন্তু সংকলন হবে না, ত্রুটি দেখাবে, কারণ হলো পূর্ব ও উত্তর বৃদ্ধির মধ্যে উত্তর বৃদ্ধির অগ্রগণ্যতা (precedence) আগে, ফলে ++x++ আসলে ++(x++) এর সমতুল। অগ্রগণ্যতার ক্রমের (precedence order) নিয়মগুলো আমরা পরের এক পাঠে বিস্তারিত জানবো।

এবারে আমরা বৃদ্ধি ব্যবহারে ক্রমলেখয়ের গতির ওপর প্রভাব নিয়ে একটু আলোচনা করি। বৃদ্ধি (increment) ++x বা x++ সাধারণত x+=1 বা x=x+1 এর চেয়ে দ্রুতগতির, এর কারণ মুলত একদম যন্ত্র পর্যায়ে x++ বা ++x বিশেষ ভাবে নির্বাহিত হয় কিন্তু x+=1 বা x=x+1 সাধারণ যোগের মতো করে নির্বাহিত হয়। সাধারণতো পূর্ব-বৃদ্ধি (pre-increment) আর উত্তর-বৃদ্ধির (post-increment) মধ্যে পূর্ব-বৃদ্ধি দ্রুত গতির। কারণ হলো, উত্তর-বৃদ্ধির ফলাফল যেহেতু x এর মান বৃদ্ধি করবার আগের মান, তাই ওই আগের মানটি প্রথমে কোথাও ক্ষণস্থায়ী (temporarily) ভাবে রেখে দিতে হয়, আর x এর মান বৃদ্ধিটা তারপর ঘটে, আর তারপর ক্ষণস্থায়ী ভাবে রাখা মানটা ফলাফল হিসাবে আসে যেটি y = x++; এর ক্ষেত্রে y এ আরোপিত হয়। কিন্তু পূর্ব-বৃদ্ধির ক্ষেত্রে মান বৃদ্ধি আগে ঘটে আর ফলাফলটাও সেই বৃদ্ধিপ্রাপ্ত মানই, কাজেই ক্ষণস্থায়ী ভাবে আগের মান রেখে দেওয়ার কোন বোঝা (overhead) এখানে নেই। মোটকথা পূর্ব-বৃদ্ধি সরাসরি বাম-মানের ওপরই কাজ করে অর্থাৎ ++x এ সরাসরি চলকটার ওপরই কাজ করে, আর কোন ক্ষণস্থায়ী কিছু দরকার হয় না। এ কারণে পূর্ব-বৃদ্ধি ++x; উত্তর-বৃদ্ধি x++; এর চেয়ে বেশী দ্রুতগতির হয়ে থাকে। কাজেই তুমি পারতো পক্ষে ++x ব্যবহার করবে, x++ ব্যবহার করবে না।

ক্রমলেখতে বৃদ্ধি ব্যবহারে এবারে একটা পরামর্শ দেই। পূর্ব-বৃদ্ধি ও উত্তর-বৃদ্ধি নিয়ে অনেক রকম খেলা যায়, যেমন তুমি চাইলে x = (++x)++ + ++x; এর মতো অনেকগুলো + চিহ্ন দিয়ে কিছু একটা লিখতে পারো। এই রকম জটিল বিবৃতিগুলো হয়তো সংকলন (compile) হবে। এর ফলে ফলাফলও কিছু একটা আসবে, যেটা চাইলে বুঝা সম্ভব, কিন্তু বুঝতে গেলে মাথা বেশ গরম হয়ে যায়। আমার পরামর্শ হলো এইরকম জটিল বিবৃতি পারতো পক্ষে লেখবে না। সবসময় এমন ভাবে সংকেত (code) লিখবে যাতে পরে তুমি বা অন্য কেউ তেমন কোন কষ্ট ছাড়াই তোমার সংকেত দেখে বুঝতে পারে। মনে রাখবে সংকেত যত জটিল, তার ভুল বের করাও তত কঠিন।

উপরের পুরো আলোচনাতে আমরা কেবল বৃদ্ধি (increment) নিয়ে আলোচনা করেছি। আসলে হ্রাস (decrement) −− নিয়ে আলোচনাটা একদম একই রকম। আমরা তাই পুনরাবৃত্তি করবো না। কেবল জেনে রাখো হ্রাসের (decrement) ফলে মান 1 কমে যায়। তাই −−x বা x−− হলো x −= 1 বা x = x − 1 এর সমতুল। আমরা −−x কে পূর্ব হ্রাস (pre-decrement) আর x−− কে উত্তর-হ্রাস (post-decrement) বলি। পূর্ব-হ্রাসের তুলনায় উত্তর-হ্রাসের অগ্রগণ্যতা (precedence) বেশী। গতির দিক বিবেচনায় পূর্ব-হ্রাস, উত্তর-হ্রাসের চেয়ে শ্রেয়তর।

৫.৭ বির্তি অণুক্রিয়া (Comma Operator)

সিপিপিতে বির্তি অণুক্রিয়া (comma operator) কয়েকটি রাশি (expression) কে এক সাথে পরপর লেখায় সাহায্য করে। বির্তি (comma) অণুক্রিয়ার বামপাশের উপাদানের (operand) মান সব সময় নর্থক (void) হয় আর উপেক্ষিত হয়। এর অর্থ হচ্ছে ডান পাশের উপাদানটির (operand) মানই বির্তি অণুক্রিয়ার (comma operator) ফলাফল হয়।

একটা উদাহরণ দেখি x = (y=3, y+1); এই বিবৃতির ফলে বন্ধনীর ভিতরে প্রথমে বির্তির বাম পাশের রাশি হিসাবে y এর মান আরোপিত (assign) হবে 3। যদিও আরোপনের কারণে আমরা y এ 3 আরোপণের পাশাপাশি ফলাফলও পাই 3, কিন্তু বির্তির (comma) কারণে সেই

ফলাফল বাদ গিয়ে ফলাফল হয়ে যাবে নর্থক (void)। যাইহোক এরপর বির্তির (comma) ডান পাশের রাশি হিসাবে y+1 এর মান 3+1 বা 4 হবপ যেটি আসলে যোগেরও + ফলাফল। আর যোগের এই ফলাফল 4 ই শেষ পর্যন্ত x চলকে আরোপিত হবে। এখানে বন্ধনী দরকার কারণ বির্তি (comma) , সাধারণত আরোপণ (assignment) = এর পরে হিসাব করা হয়। আমরা বন্ধনীর ভিতরের আরোপণটি y = 3 বির্তির (comma) আগে করতে চাইলেও বন্ধনীর বাইরের চলক x এ আরোপণটি বির্তির পরে করতে চাই, আর এ কারণে বন্ধনী জরুরী। ব্যাপারটি আরো পরিস্কার বুঝতে চাইলে একই জিনিস বন্ধনী ছাড়া কী হবে দেখো `x = y = 3, y + 1;`। এখানে দুটো আরোপণই (assignment) বির্তির (comma) আগে নির্বাহিত (execute) হবে। ফলে প্রথমে y এর মান আরোপিত হবে 3, তারপর x এও মান 3ই আরোপিত হবে, তারপর y+1 হিসাব হবে 4। এই 4 বির্তির ফলাফল হলেও সেটি কিন্তু এখানে কিছুতে আরোপিত হয় নি।

বির্তি (comma) অণুক্রিয়া (operator) হিসাবে ব্যবহার হলেও এর আরো নানান ব্যবহার আছে সিপিপিতে। যেমন আমরা দেখেছি একাধিক চলক (variable) একসাথে ঘোষণা (declare) করতে আমরা বির্তি (comma) দিয়ে লিখি `int x, y, z = 3;` বির্তির (comma) এই রকম ব্যবহার আসলে অণুক্রিয়া হিসাবে নয়, বরং তালিকার পৃথকী (separator) হিসাবে ব্যবহার। আমরা যখন পরে ক্রম-ঘূর্ণী (for-loop) ও পরামিতি (parameter) নিয়ে আলোচনা করবো তখনও তালিকা পৃথকী (list separator) হিসাবে বির্তির (comma) ব্যবহার দেখতে পাবো।

৫.৮ অগ্রগণ্যতার ক্রম (Precedence Order)

অগ্রগণ্যতার ক্রম (precedence order) কী? সিপিপিতে এ পর্যন্ত পরিচিত হওয়া অণুক্রিয়াগুলোর (operator) অগ্রগণ্যতার ক্রম (precedence order) আলোচনা করো।

ধরো 3 + 4*5 + 6 এর মান হিসাব করবে। আগের দিনে এক রকম সস্তা কলনি (calculator) পাওয়া যেতো যেটি করতো কী, বাম থেকে হিসাব করতো একের পর এক। ফলে সেটা প্রথম 3 ও 4 যোগ করে 7 বের করতো, তারপর তার সাথে 5 গুণ করে বের করতো 35 আর শেষে তার সাথে 6 যোগ করে ফল দিতো 41। তুমি চাইলে উল্টো আরেক রকমের অবস্থা ভাবতে পারো, যেখানে ডান দিক থেকে একের পর এক হিসাব হবে। সুতরাং 5 ও 6 যোগ করে 11, তারসাথে 4 গুণ করে 44, শেষে 3 যোগ করে 47। কিন্তু ছোটবেলা থেকে সরলের নিয়ম আমরা শিখে এসেছি: গুণ আগে হবে যোগ পরে হবে। আমরা তাই হিসাব করি 4 ও 5 এর গুণ আগে ফল 20 তার সাথে বামের যোগ আগে, তাই 3 আগে যোগ হলো 23, শেষে ডানের যোগ তাই 6 যোগ করে হলো 29, যেটাকে আমরা সঠিক হিসাব বলে ধরে নেই। এই যে বাম থেকে ডানে বা ডান থেকে বামে হিসাব না করে গুণ যোগের আগে করতে হবে, আবার দুটো যোগ পর পর থাকলে বামের যোগ আগে করতে হবে। এই নিয়মগুলোকে অগ্রগণ্যতার ক্রম(precedence order) বলা হয়।

সরল অংকে অগ্রগণ্যতার ক্রম ছিল: বন্ধণী, এর, ভাগ, গুণ, যোগ, বিয়োগ। সবেচেয়ে ভিতরের বন্ধনী সবেচেয়ে আগে। ভাগ আর গুণ আসলে বাম থেকে যেটা আগে আসে। একই ভাবে যোগ ও বিয়োগ বাম থেকে যেটা আগে আসে। সিপিপিতে আমরা এ পর্যন্ত অনেকগুলো অণুক্রিয়ার (Operator) সাথে পরিচিত হয়েছি। এগুলো হলো একিক `+ − ++ −−` দুয়িক `+ − ∗ / % = += −= ∗= /− %=` , তো এদের মধ্যে একিক অণুক্রিয়ার ক্রম সবার আগে, তারপর দুয়িক অণুক্রিয়াগুলোর ক্রম। আমরা আপাতত কেবল এগুলোর অগ্রগণ্যতার ক্রম (precedence order) বিবেচনা করবো। অন্যান্য অণুক্রিয়া ও তাদের ক্রম সম্পর্কে আমরা পরে জানবো।

১. ++ -- একিক অণুক্রিয়া (unary operator) উত্তর-বৃদ্ধি ও উত্তর-হ্রাস (post-increment and post-decrement) x++, x-- এরা বাম-মানের (l- value) ওপরে প্রযুক্ত হয়ে

ডান-মান (r-value) ফলাফল দেয়। ফলে x++++ বা x---- করা যায় না।

২. ++ -- + - একিক অণুক্রিয়া (unary operator) পূর্ব-বৃদ্ধি (pre-increment) ++x ও পূর্ব-হ্রাস (pre-decrement) --x এরা বাম-মানের (l-value) ওপর প্রযুক্ত হয়ে বাম-মানই ফলাফল দেয়। ফলে ++++x বা ----x করা যায়, আর সবেচেয়ে ডানের ++ বা -- আগে প্রযুক্ত হয়। (পূর্ব) একিক অণুক্রিয়া (unary operator) +x ধনাত্মক (positive) -x আর ঋণাত্মক (negative) এরা ডান-মানের (r-value) ওপর প্রযুক্ত হয়ে ডান-মানই দেয়। ফলে + + x বা - - x করা সম্ভব, খেয়াল করো দুটো + বা দুটো - এর মধ্যে ফাঁকা দিতে হয়েছে না হলে ওগুলো বৃদ্ধি বা হ্রাস হিসাবে চিহ্নিত হয়ে যাবে।

৩. * / % দুয়িক অণুক্রিয়া (binary operator) এরা দুটি ডান-মানের (r-value) উপাদানের (operand) ওপর প্রযুক্ত হয়ে ডান-মানই ফল দেয়। এই অণুক্রিয়াগুলো পরপর অনেকগুলো থাকলে বাম থেকে ডানে একে একে হিসাব হতে থাকে। যেমন 10 / 2 * 4 % 6 এ বাম থেকে ডানে প্রথমে ভাগফল, তারপর গুণফল, তারপর ভাগশেষ হিসাব হবে।

৪. + - দুয়িক অণুক্রিয়া (binary operator) এরা দুটি ডান-মানের (r-value) উপাদানের (operator) ওপর প্রযুক্ত হয়ে ডান-মানই ফল দেয়। এই অণুক্রিয়াগুলো পরপর অনেকগুলো থাকলে বাম থেকে ডানে একে একে হিসাব হতে থাকে। যেমন 10 − 2 + 5 এ বাম থেকে ডানে প্রথমে বিয়োগফল, তারপর যোগফল হিসাব হবে।

৫. = += -= += *= /= %= এই সব দুয়িক অণুক্রিয়া (binary operator) আরোপণগুলোর (assignment) বামপাশে এমন কিছু থাকতে হবে যার বাম-মান (l-value) আছে, আর ডান পাশে এমন কিছু থাকতে হয় যার ডান-মান (r-value) আছে। এই অণুক্রিয়াগুলো পরপর অনেগুলো থাকলে ডান থেকে বামে একে একে হিসাব হতে থাকে। যেমন x += y = z *= 3 তে প্রথমে ডানের *= এর কারণে z এর সাথে 3 গুণ হবে, তারপর মাঝের = এর কারণে z এর মান y আরোপিত হবে, শেষে y এর মান x এর সাথে যোগ হবে।

৬. , বির্তি (comma) একটি দুয়িক অণুক্রিয়া (binary operator) যেটির ফলাফল কেবল ডানপাশের উপাদান (operand)। বাম পাশের উপাদানটি হিসাব হয়, কিন্তু তার ফলাফল হবে নর্থক (void)। এই অণুক্রিয়া একাধিক পরপর থাকলে, বাম থেকে ডানে একে একে হিসাব হতে থাকে। যেমন x + 2, y * 3, z / 4 প্রথমে যোগ হবে, তারপর গুণ আর শেষে ভাগ, ফলাফল হবে একদম ডানের ভাগফলটিই।

দুটো একই বা একই ক্রমের অণুক্রিয়া পরপর থাকলে কোন পাশেরটি আগে হবে এইটি নির্ধারণ করে দেয়াকে বলা হয় সহযোজ্যতা (associativity)। যেমন x-y-z থাকলে আমাদের প্রথমে বামের বিয়োগ করতে হবে, তারপর ডানের বিয়োগ, কাজেই বিয়োগ হল বাম সহযোজ্য (left associative) অর্থাৎ x - y - z আর (x -y) - z একই। খেয়াল করো বিয়োগ কিন্তু ডান সহযোজ্য (right associative) নয় কারণ x - y - z আর x - (y - z) এক নয়। যোগ আবার বাম ও ডান উভয় সহযোজ্য কারণ x + y + z, (x + y) + z ও x + (y + z) একই। সাধারণত উভয় সহযোজ্যদের ক্ষেত্রে সুবিধার্থে তাদের বাম-সহযোজ্য হিসাবে বিবেচনা করা হয়। উপরের তালিকায় আলোচিত অণুক্রিয়াগুলোর ক্ষেত্রে একই রকম অণুক্রিয়া পরপর থাকলে কোন পাশেরটি আগে হবে, সেটাও কিন্তু আলোচনা করা হয়েছে। সেখান থেকে বুঝতো পারো কোন অণুক্রিয়া বাম সহযোজ্য (left associative), আর কোনটি ডান সহযোজ্য (right associative)?

সবেশেষে একটা গুরুত্বপূর্ণ বিষয় মনে রাখবে বন্ধনী () এর শক্তি কিন্তু সবচেয়ে বেশী। যে কোন স্থানে কোন রকমের দ্বিধাদ্বন্দ্ব থাকলে সেখানে বন্ধনী ব্যবহার করে দ্বিধা পরিস্কার করবে। অণুক্রিয়াগুলোর (operator) অগ্রগণ্যতার ক্রম (precedence order) ব্যবহার করে নানা রকম

জটিল জটিল বিবৃতি ও রাশি (statement and expression) তৈরী করা যায়, যেগুলো ক্রম বিবেচনায় নিয়ে বুঝতে গেলে মাথা গরম হয়ে যেতে পারে, ভুল হলে বের করা কঠিন হয়ে যাবে। কাজেই আমার পরামর্শ হচ্ছে তোমার বিবৃতি বা রাশি অবশ্যই সহজে পাঠযোগ্য হতে হবে, আর এ কাজে যত দরকার বন্ধনী ব্যবহার করবে। যেমন ধরো x += y − z এর চেয়ে x += (y − z) বুঝা আমাদের জন্য বেশী সহজ, কারণ এতে একদম পরিস্কার বিয়োগ আগে হবে।

৫.৯ গাণিতিক সমস্যা (Mathematical Problems)

দ্বিমাত্রিক স্থানাঙ্ক ব্যবস্থায় (two dimensional coordinate system) দুটি বিন্দুর স্থানাঙ্ক যোগান (input) নিয়ে তাদের মাঝে দূরত্ব ফলন (output) হিসাবে দেখাও। ধরো স্থানাঙ্কগুলো ভগ্নকে দেয়া আছে। তোমার নিশ্চয় জানা আছে যে দুটো বিন্দুর (x_1, y_1) ও (x_2, y_2) দূরত্ব হলো $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ অর্থাৎ ভুজদ্বয়ের দূরত্বের বর্গ ও কোটিদ্বয়ের দূরত্বের বর্গের যোগফলের বর্গমূল। বর্গমূল নির্ণয়ের জন্য cmath শির নথি (header file) থেকে sqrt বিপাতক ব্যবহার করো। আর বর্গ নির্ণয়ের জন্য তোমাকে একই জিনিস দুইবার গুণ করতে হবে।

ফিরিস্তি ৫.৫: দুটি বিন্দুর মধ্যের দূরত্ব (Distance Between Two Points)

```
// নীচের শির নথি main বিপাতকের বাইরে অন্তর্ভুক্ত করো

#include <cmath>  //  বর্গমূল নির্ণয়ের জন্য  sqrt বিপাতক লাগবে

// নীচের অংশ main বিপাতকের ভিতরে return এর আগে লিখো

float x1, y1, x2, y2; // বিন্দুদুটোর স্থানাঙ্ক (x1,y1), (x2,y2)

cout << "prothom bindur x y: ";      // যোগান যাচনা
cin >> x1 >> y1;                     // প্রথম বিন্দু যোগান
cout << "ditiyo bindur x y: ";       // যোগান যাচনা
cin >> x2 >> y2;                     // দ্বিতীয় বিন্দু যোগান

float xd = abs(x1 - x2);                 // ভুজ দুটির দূরত্ব
float yd = abs(y1 - y2);                 // কোটি দুটির দূরত্ব

float dd = sqrt(xd * xd + yd * yd); // দূরত্ব হিসাব করো
cout << "bindu dutor durotto: " << dd << endl;  // ফলন
```

উপরের ক্রমলেখ খেয়াল করো। খুবই সাদামাটা। প্রথমে main বিপাতকের বাইরে cmath শির নথি অন্তর্ভুক্ত করতে হবে বলে দেখানো হয়েছে। তারপর main বিপাতকের ভিতরে বিন্দু দুটোর ভুজ ও কোটি ধারণ করার জন্য চারটি float ধরনের ভগ্নক চলক (variable) নেয়া হয়েছে। এরপর যোগান যাচনা (input prompt) দিয়ে বিন্দুদুটোর স্থানাঙ্ক যোগান (input) নেয়া হয়েছে। তারপর ভুজ দ্বয়ের দূরত্ব abs(x1 - x2) বের করে xd নামের আরেকটি চলকে নেয়া হয়েছে, একই ভাবে কোটিদ্বয়ের দূরত্ব abs(y1 - y2) বের করে yd নামের আরেকটি চলকে নেয়া হয়েছে। মনে করে দেখো abs বিপাতকটি (function) কোন সংখ্যার পরম মান (absolute value) অর্থাৎ

চিহ্ন বাদ দিয়ে কেবল মানটুকু ফেরত দেয়। যাইহোক তারপর xd এর বর্গ ও yd এর বর্গের যোগফল নিয়ে তার বর্গমূল বের করা হয়েছে sqrt বিপাতক ব্যবহার করে আর রাখা হয়েছে dd চলকে। সবশেষে দূরত্ব dd চলক থেকে ফলন (output) দেয়া হয়েছে। এখানে একটা কথা বলে রাখি sqrt(xd * xd + yd * yd) এর বদলে cmath শির নথি (header file) থেকেই hypot নামের বিপাতকও (function) আমরা ব্যবহার করতে পারতাম। সেক্ষেত্রে আমাদের লিখতে হতো hypot(xd, yd) আর সেটি ঠিক একই কাজ করতো।

৫.১০ শির নথি cmath (Header File cmath)

শির নথি cmath এ গাণিতিক প্রক্রিয়াকরণে ব্যবহৃতব্য নানান বিপাতক (function) আছে। আমরা এখানে ওই বিপাতকগুলোর সাথে সংক্ষিপ্ত আকারে পরিচিত হবো। এই বিপাতকগুলো কী তা বুঝতে তোমার উচ্চমাধ্যমিক গণিতের ধারণাবলী দরকার হবে। নীচের পরাবৃত্তীয় (hyperbolic) বিপাতকগুলো ছাড়া প্রায় সবগুলো বিপাতকই আমাদের প্রায়শই কাজে লাগে।

গাণিতিক বিপাতক (Mathematical Functions)

- abs(x): কোন সংখ্যা x এর পরম মান। abs(3) হবে 3 এবং abs(-3) হবে 3।

ত্রিকোণমিতিক বিপাতক (Trigonometric Functions)

- cos(x): লগ্নানুপাত (cosine) যেখানে x হল রেডিয়ানে।
- sin(x): লম্বানুপাত (sine) যেখানে x হল রেডিয়ানে।
- tan(x): স্পর্শানুপাত (tangent) যেখানে x হল রেডিয়ানে।
- acos(x): বিলগ্নানুপাত (arc-cosine) যেখানে ফেরত মান রেডিয়ানে।
- asin(x): বিলম্বানুপাত (arc-sine) যেখানে ফেরত মান রেডিয়ানে।
- atan(x): বিস্পর্শানুপাত (arc-tangent) যেখানে ফেরত মান রেডিয়ানে।
- atan2(x,y): বিস্পর্শানুপাত (arc-tangent) যেখানে $\frac{x}{y}$ এর x হল লব (numerator) আর y হল হর (denominator) আর ফেরত মান রেডিয়ানে।

পরাবৃত্তীয় বিপাতক (Hyperbolic Functions)

- cosh(x): পরাবৃত্তীয় লগ্নানুপাত (hyperbolic cosine) যেখানে x হল রেডিয়ানে।
- sinh(x): পরাবৃত্তীয় লম্বানুপাত (hyperbolic sine) যেখানে x হল রেডিয়ানে।
- tanh(x): পরাবৃত্তীয় স্পর্শানুপাত (hyperbolic tangent) যেখানে x হল রেডিয়ানে।
- acosh(x): পরাবৃত্তীয় বিলগ্নানুপাত (hyperbolic arc-cosine), ফেরত রেডিয়ানে।
- asinh(x): পরাবৃত্তীয় বিলম্বানুপাত (hyperbolic arc-sine), ফেরত রেডিয়ানে।

- atanh(x): পরাবৃত্তীয় বিস্পর্শানুপাত (hyperbolic arc-tangent), ফেরত রেডিয়ানে।

সুচক ও ঘাতাঙ্ক (Exponents and Logarithms)

- exp(x): e^x বা সুচকীয় বিপাতক (exponential function)
- log(x): $\log_e x$ বা ঘাতাঙ্ক বিপাতক (logarithmic function)
- log10(x): $log_{10}x$ বা ১০-ভিত্তিক ঘাতাঙ্ক (logarithm)
- exp2(x): 2^x বা ২-ভিত্তিক সুচকীয় (exponential) বিপাতক
- log2(x): log_2x বা ২-ভিত্তিক ঘাতাঙ্ক (logarithm)

শক্তি ও ঘাত (Power and Index)

- pow(x,y): x^y অর্থাৎ x এর y তম শক্তি যেমন pow(2,3) হল 2^3 বা 8
- sqrt(x): $\sqrt{x}$ অর্থাৎ x এর বর্গমূল যেমন sqrt(16.0) হল 4.0
- cbrt(x): $\sqrt[3]{x}$ অর্থাৎ x এর ঘনমূল যেমন cbrt(8.0) হল 2.0
- hypot(x,y): $\sqrt{x^2+y^2}$ অর্থাৎ x ও y কে সমকোনী ত্রিভুজের লম্ব (perpendicular) ও ভুমি (base) ধরলে অতিভুজের (hypotenuse) দৈর্ঘ্য

নৈকটায়নের বিপাতক (Rounding Functions)

- round(x): নৈকটায়ন বিপাতক x এর নিকটতম পূর্ণক।
- floor(x): মেঝে বিপাতক x এর সমান বা ঠিক ছোট পূর্ণকটি।
- ceil(x): ছাদ বিপাতক x এর সমান বা ঠিক বড় পূর্ণকটি।
- trunc(x): কর্তন বিপাতক x এর ভগ্নাংশটুকু কেটে ফেলবে।

উপরের বিপাতকগুলোর ফলাফল বুঝার জন্য নীচের সারণী লক্ষ্য করো।

মান x	নৈকটায়ন round(x)	মেঝে floor(x)	ছাদ ceil(x)	কর্তন trunc(x)
2.3	2.0	2.0	3.0	2.0
2.8	3.0	2.0	3.0	2.0
2.5	3.0	2.0	3.0	2.0
2.0	2.0	2.0	2.0	2.0
-2.3	-2.0	-3.0	-2.0	-2.0
-2.8	-3.0	-3.0	-2.0	-2.0
-2.5	-3.0	-3.0	-2.0	-2.0

৫.১১ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. বিপাতক (function) ও রাশি (expression) বলতে কী বুঝো? উদাহরণ দাও।

২. একিক (unary) ও দুয়িক (binary) অণুক্রিয়া (operation) বলতে কী বুঝো? কয়েকটা করে একিক (unary) ও দুয়িক (binary) অণুক্রিয়ার (operation) নাম বলো।

৩. উপাত্ত প্রকারান্তর (type casting) কী? দুয়িক অণুক্রিয়ায় (binary operation) কী ভাবে উপাত্ত প্রকারান্তর (type casting) হয়?

৪. নির্বাহ-কালীন ত্রুটি (execution-time error) বলতে কী বুঝো? ভাগফল ও ভাগশেষ নির্ণয়ের সময় কোন নির্বাহকালীন ত্রুটি ঘটতে পারে?

৫. অদরকারী সংকেতাংশ (code segment) মুছে না দিয়ে কী ভাবে আমরা টীকা (comment) ব্যবহার করে সেগুলোকে অকার্যকর করে রাখতে পারি, ব্যাখ্যা করো।

৬. ঋণাত্মক পূর্ণকের (integer) ভাগফল ও ভাগশেষ নির্ণয়ের নিয়ম বর্ণনা করো।

৭. আরোপণ অণুক্রিয়ার (assignment operator) ফলাফল কী? যৌগিক আরোপণ (compound assignment) বলতে কী বুঝো? কয়েকটি যৌগিক আরোপণের উদাহরণ দাও।

৮. সাধারণ যৌগিক আরোপন (compound assignment) যেমন x += 1 ব্যবহার না করে কেন বৃদ্ধি (increment) x++ বা ++x কেন ব্যবহার করা হয়?

৯. উত্তর-বৃদ্ধি (post-increment) ও পূর্ব-বৃদ্ধি (pre-increment) এর মধ্যে পার্থক্যগুলো আলোচনা করো। তুমি কোনটি ব্যবহার করতে চাইবে এবং কেন?

১০. বির্তি (comma) অণুক্রিয়ার কাজ কী? এর ফলাফলই বা কী?

১১. অগ্রগণ্যতার ক্রম (precedence order) ও সহযোজ্যতা (associativity) কী?

১২. সিপিপিতে এ পর্যন্ত তোমার জানা অণুক্রিয়াগুলোর (operator) অগ্রগণতার ক্রম (precedence order) ও সহযোজ্যতা (associativity) আলোচনা করো।

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. একটি সমান্তর ধারার (arithmetic series) প্রথম পদ a সাধারণ অন্তর d হলে n-তম পদ কতো? n পদের সমষ্টিই বা কত? এর জন্য সিপিপিতে একটা ক্রমলেখ (program) তৈরী করো যেটি a, d, ও n যোগান (input) নিবে, আর n-তম পদ ও n পদের সমষ্টি ফলন (output) দিবে। এর জন্য তুমি সূত্র ব্যবহার করবে n-তম পদ = $a + (n-1) * d$ আর n পদের সমষ্টি = $n * (2a + (n-1) * d)/2$। প্রদত্ত বিভিন্ন ধারার জন্যে এই সূত্র a আর d বসালে আমরা ওই ধারাগুলোর জন্য সরাসরি সূত্র পেতে পারি।

- 1 + 2 + 3 + ... ধারাতে a = 1, d = 1। সুতরাং n-তম পদ = $a + n - 1$, n পদের সমষ্টি = $n(n + 1)/2$। যেমন n = 10 হলে 10-তম পদ 10, সমষ্টি 55।
- 2 + 4 + 6 + ... ধারাতে a = 2, d = 2। সুতরাং n-তম পদ = $2n$, n পদের সমষ্টি = $n(n + 1)$। যেমন n = 10 হলে 10-তম পদ 20, সমষ্টি 110।
- 1 + 3 + 5 + ... ধারাতে a = 1, d = 2। সুতরাং n-তম পদ = $2n - 1$, n পদের সমষ্টি = n^2। যেমন n = 10 হলে 10-তম পদ 19, সমষ্টি 100।

২. নীচের মতো ফলন (output) দেয় এরকম একটি ক্রমলেখ (program) তৈরী করো। ফলের স্তম্ভটিতে তুমি দুয়িক অণুক্রিয়াগুলো (binary operator) ব্যবহার করবে।

```
x=10  y=5

rashi  fol
x=y+3  x=  8
x=y-2  x=  3
x=y*5  x=  25
x=x/y  x=  2
x=x%y  x=  0
```

৩. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি তিন অঙ্কের সংখ্যাকে উল্টো করে যেমন 326 হয়ে যায় 623। এ কাজে তুমি ভাগফল, ভাগশেষ, গুণ, যোগ ও বিয়োগ ব্যবহার করবে। 326 থেকে অঙ্কগুলো আলাদা করে তারপর 623 তৈরী করবে।

৪. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c যোগান (input) নিয়ে ত্রিভুজটির ক্ষেত্রফল নির্ণয় করো। তুমি হয়তো জানো ত্রিভুজের ক্ষেত্রফল = $\sqrt{s(s-a)(s-b)(s-c)}$ যেখানে s হলো অর্ধ পরিসীমা অর্থাৎ $s = (a + b + c)/2$।

৫. এমন একটি ক্রমলেখ (program) রচনা করো যেটি সেকেন্ড যোগান নিয়ে তাকে ঘন্টা-মিনিট-সেকেন্ডে রূপান্তর করে। এ কাজে তুমি ভাগফল ও ভাগশেষ ব্যবহার করবে।

৬. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c যোগান (input) নিয়ে এর কোণগুলো নির্ণয় করো। ধরো ত্রিভুজের কোন তিনটি A, B, C। এখান A, B, C যথাক্রমে a, b, c বাহুর বিপরীত কোণ। তুমি হয়তো জানো কোণ $C = \cos^{-1}((a^2 + b^2 - c^2)/(2ab))$, কোণ $B = \cos^{-1}((c^2+a^2-b^2)/(2ca))$ ও কোণ $A = \cos^{-1}((b^2+c^2-a^2)/(2bc))$। তোমার ক্রমলেখতে ত্রিভুজের কোনগুলোকে তুমি ডিগ্রীতে রূপান্তর করে ফলন দিবে।

৭. এমন একটি ক্রমলেখ (program) রচনা করো যেটি দুটো সময় ঘন্টা, মিনিট, সেকেন্ডে নিয়ে সময় দুটিকে যোগ করে। এ কাজে তুমি যোগ, ভাগফল ও ভাগশেষ ব্যবহার করবে।

৮. এমন একটি ক্রমলেখ রচনা করো যেটি দুটো সমীকরণ $ax + by = c$ ও $dx + ey = f$ এর a, b, c, d, e, f যোগান নিয়ে x ও y এর মান ফলন দেয়।

৯. একটি বাস u আদিবেগ ও a সমত্বরণ নিয়ে যাত্রা শুরু করলো। সময় t সেকেন্ড পরে বাসের গতিবেগ v নির্ণয় করো। t সময় পরে বাসটি অতিক্রান্ত দুরত্ব sও নির্ণয় করো। এ কাজে তুমি গতিবিদ্যার সূত্র $v = u + at$ ও $s = ut + \frac{1}{2}at^2$ ব্যবহার করবে।

৫.১১. অনুশীলনী সমস্যা (Exercise Problems)

১০. নীচের ছদ্ম-সংকেতের (pseudocode) জন্য একটি ক্রমলেখ (program) তৈরী করো।

ক) পড়ো (read) x ও y

খ) গণো (compute) $p = x * y$

গ) গণো (compute) $s = x + y$

ঘ) গণো (compute) $t = s^2 + p * (s - x) * (p + y)$

ঙ) লিখো (write) t

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. একটি সমান্তর ধারার (arithmetic series) প্রথম পদ a সাধারণ অন্তর d হলে n-তম পদ কতো? n পদের সমষ্টিই বা কত? এর জন্য সিপিপিতে একটা ক্রমলেখ (program) তৈরী করো যেটি a, d, ও n যোগান (input) নিবে, আর n-তম পদ ও n পদের সমষ্টি ফলন (output) দিবে। এর জন্য তুমি সূত্র ব্যবহার করবে n-তম পদ = $a + (n-1) * d$ আর n পদের সমষ্টি = $n * (2a + (n-1) * d)/2$। প্রদত্ত বিভিন্ন ধারার জন্যে এই সূত্র a আর d বসালে আমরা ওই ধারাগুলোর জন্য সরাসরি সূত্র পেতে পারি।

- 1 + 2 + 3 + ... ধারাতে a = 1, d = 1। সুতরাং n-তম পদ = $a + n - 1$, n পদের সমষ্টি = $n(n+1)/2$। যেমন n = 10 হলে 10-তম পদ 10, সমষ্টি 55।
- 2 + 4 + 6 + ... ধারাতে a = 2, d = 2। সুতরাং n-তম পদ = $2n$, n পদের সমষ্টি = $n(n+1)$। যেমন n = 10 হলে 10-তম পদ 20, সমষ্টি 110।
- 1 + 3 + 5 + ... ধারাতে a = 1, d = 2। সুতরাং n-তম পদ = $2n - 1$, n পদের সমষ্টি = n^2। যেমন n = 10 হলে 10-তম পদ 19, সমষ্টি 100।

আমরা এখানে কেবল সাধারণ সুত্রের জন্য ক্রমলেখ (program) তৈরী করবো। প্রদত্ত বিশেষ ধারার জন্য তুমি এই ক্রমলেখ (program) দরকার মতো বদলে নিতে পারবে।

ফিরিস্তি ৫.৬: সমান্তর ধারার সমস্যা (Arithmetic Series Problem)

```
int a, d, n;
cout << "prothom pod? "; cin >> a;
cout << "sadharon ontor? "; cin >> d;
cout << "kototom pod?"; cin >> n;

int t = a + (n - 1) * d;  //  n-তম পদ
cout << n << "-tom pod = " << t << endl;

int s = n * (2*a + (n - 1)*d) / 2; // সমষ্টি
cout << n << " poder somosti = " << s << endl;
```

যোগান-ফলনাংশ (input-output segment)

```
prothom pod? 1
sadharon ontor? 1
kototom pod? 10
10-tom pod = 10
10 poder somosti = 55
```

২. নীচের মতো ফলন (output) দেয় এরকম একটি ক্রমলেখ (program) তৈরী করো।

```
x=10 y=5

rashi fol
x=y+3 x= 8
x=y-2 x= 3
x=y*5 x= 25
x=x/y x= 2
x=x%y x= 0
```

ফিরিস্তি ৫.৭: দুয়িক অণুক্রিয়ার ফলাফল (Binary Operation Results)

```
int x = 10, y = 5;

cout << "x=" << x << " y=" << y << endl;
cout << endl; //  ফাঁকা সারি
cout << "rashi "<< "fol " << endl;
cout << "x=y+3" << " x= " << y+3 << endl;
cout << "x=y-2" << " x= " << y-2 << endl;
cout << "x=y*5" << " x= " << y*5 << endl;
cout << "x=x/y" << " x= " << x/y << endl;
cout << "x=x%y" << " x= " << x%y << endl;
```

৩. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি তিন অঙ্কের সংখ্যাকে উল্টো করে যেমন 326 হয়ে যায় 623। এ কাজে তুমি ভাগফল, ভাগশেষ, গুণ, যোগ ও বিয়োগ ব্যবহার করবে। 326 থেকে অঙ্কগুলো আলাদা করে তারপর 623 তৈরী করবে।

```
int soja = 326;

int daner = soja % 10;  // ভাগশেষ 6
int bamer = soja / 100; // ভাগফল 3
int majher = soja / 10 % 10; //ফল 2

int ulta = bamer;      // উল্টা = 3
ulta += majher * 10;   // উল্টা = 23
ulta += daner * 100;   // উল্টা = 623
```

৫.১১. অনুশীলনী সমস্যা (Exercise Problems)

৪. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c যোগান (input) নিয়ে ত্রিভুজটির ক্ষেত্রফল নির্ণয় করো। তুমি হয়তো জানো ত্রিভুজের ক্ষেত্রফল = $\sqrt{s(s-a)(s-b)(s-c)}$ যেখানে s হলো অর্ধ পরিসীমা অর্থাৎ $s = (a+b+c)/2$।

ফিরিস্তি ৫.৮: ত্রিভুজের বাহু হতে ক্ষেত্রফল (Triangle's Area From Sides)

```
// main বিপাতকের বাইরে
#include <cmath>

// main বিপাতকের ভিতরে
float a, b, c;                  // বাহুগুলো
cout << "sides a b c: ";  // যোগান যাচনা
cin >> a >> b >> c;          // যোগান নেওয়া

float s = (a + b + c) / 2;  // অর্ধ পরিসীমা
float k = sqrt(s*(s-a)*(s-b)*(s-c));  // ক্ষেত্রফল

cout << "khetrofol = " << k << endl;  // ফলন
```

যোগান-ফলনাংশ (input-output segment)

```
sides a b c: 100 60 90
khetrofol = 2666
```

৫. এমন একটি ক্রমলেখ (program) রচনা করো যেটি সেকেন্ড যোগান নিয়ে তাকে ঘন্টা-মিনিট-সেকেন্ডে রূপান্তর করে। এ কাজে তুমি ভাগফল ও ভাগশেষ ব্যবহার করবে।

ফিরিস্তি ৫.৯: সময়কে সেকেন্ডে প্রকাশ (Time in Seconds)

```
int motsekend = 38185;

int sekend = motsekend % 60; // ফল  25
int motminit = motsekend / 60;  // ফল 636

int minit = motminit % 60;  // ফল 36
int ghonta = motminit / 60; // ফল 10
```

৬. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c যোগান (input) নিয়ে এর কোণগুলো নির্ণয় করো। ধরো ত্রিভুজের কোন তিনটি A, B, C। এখান A, B, C যথাক্রমে a, b, c বাহুর বিপরীত কোণ। তুমি হয়তো জানো কোণ $C = \cos^{-1}((a^2+b^2-c^2)/(2ab))$, কোণ $B = \cos^{-1}((c^2+a^2-b^2)/(2ca))$ ও কোণ $A = \cos^{-1}((b^2+c^2-a^2)/(2bc))$। তোমার ক্রমলেখতে ত্রিভুজের কোনগুলোকে তুমি ডিগ্রীতে রূপান্তর করে ফলন দিবে।

আমরা cmath শির নথি থেকে বিলগ্নানুপাতের (arccosine) জন্য acos বিপাতকটিকে (function) ব্যবহার করবো। কিন্তু এটি আমাদের রেডিয়ানে কোণ ফেরত দিবে। রেডিয়ান থেকে ডিগ্রীতে নিতে চাইলে আমাদের 180/pi দিয়ে গুণ করতে হবে। কথা হচ্ছে পাই

কেমনে পাবো। আমরা pai একটা ধ্রুবক ঘোষণা করতে পারি যার মান দিয়ে দিব 3.1416 অথবা আরো নিখুত মান পেতে চাইলে acos(-1) থেকেও মান বের করে নিতে পারি।

ফিরিস্তি ৫.১০: ত্রিভুজের বাহু হতে কোণ (Triangle's Angles From Sides)

```
// main বিপাতকের বাইরে
#include <cmath>

// main বিপাতকের ভিতরে
float a, b, c;                  //  বাহুগুলো
cout << "sides a b c: ";  // যোগান যাচনা
cin >> a >> b >> c;          // যোগান নেওয়া

//  কোণ নির্ণয় রেডিয়ানে
float C = acos((a*a + b*b - c*c)/(2*a*b));
float B = acos((c*c + a*a - b*b)/(2*c*a));
float A = acos((b*b + c*c - a*a)/(2*b*c));

//  ডিগ্রীতে রূপান্তর
float const pai = arccos(-1); // বিকল্প হলো 3.1416
C *= 180/pai; B *= 180/pai; A *= 180/pai;

cout << "angles A B C= "; // ফলন
cout << A << " " << B << " " << C << endl;
```

যোগান-ফলনাংশ (input-output segment)

```
sides a b c: 145 60 90
angles A B C= 149.703 12.049 18.2475
```

৭. এমন একটি ক্রমলেখ (program) রচনা করো যেটি দুটো সময় ঘন্টা, মিনিট, সেকেন্ডে নিয়ে সময় দুটিকে যোগ করে। এ কাজে তুমি যোগ, ভাগফল ও ভাগশেষ ব্যবহার করবে।

ফিরিস্তি ৫.১১: সময়কে সেকেন্ডে প্রকাশ (Time in Seconds)

```
int ghonta1, minit1, sekend1; // ১ম সময় যোগান নিবে
int ghonta2, minit2, sekend2; // ২য় সময় যোগান নিবে

int sekend = sekend1 + sekend2; //  সেকেন্ড দুটো যোগ
int minit = minit1 + minit2;     // মিনিট দুটো যোগ
int ghonta = ghonta1 + ghonta2; //  ঘন্টা দুটো যোগ

minit += sekend / 60; // মোট সেকেন্ড 60 এর বেশী হলে
sekend = sekend % 60; // মিনিট হওয়ার পরে অবশিষ্ট সেকেন্ড
ghonta += minit / 60; // মোট মিনিট 60 এর বেশী হলে
minit = minit % 60;    // ঘন্টা হওয়ার পরে অবশিষ্ট মিনিট
```

৮. এমন একটি ক্রমলেখ রচনা করো যেটি দুটো সমীকরণ $ax+by=c$ ও $dx+ey=f$ এর a, b, c, d, e, f যোগান নিয়ে x ও y এর মান ফলন দেয়। এরকম সহ সমীকরণ সমাধানের সূত্র হল $x=(ce-bf)/(ae-bd)$ আর $y=(af-cd)/(ae-bd)$।

ফিরিস্তি ৫.১২: সহ সমীকরণ সমাধান (Simultaneous Equations)

```
float a, b, c, d, e, f;

cout << "prothom somikoron a b c:";
cin >> a >> b >> c;

cout << "ditiyo somikoron e f g:";
cin >> d >> e >> e;

float x = (c*e - b*f)/(a*e - b*d);
float y = (a*f - c*d)/(a*e - b*d);

cout << "x = " << x << " ";
cout << "y = " << y << endl;
```

যোগান-ফলনাংশ (input-output segment)

```
prothom somikoron a b c: 2 1 4
ditiyo somikoron e f g: 1 -1 -1
x = 1.33333 y = 1.33333
```

৯. একটি বাস u আদিবেগ ও a সমত্বরণ নিয়ে যাত্রা শুরু করলো। সময় t সেকেন্ড পরে বাসের গতিবেগ v নির্ণয় করো। t সময় পরে বাসটি অতিক্রান্ত দুরত্ব sও নির্ণয় করো। এ কাজে তুমি গতিবিদ্যার সূত্র $v=u+at$ ও $s=ut+\frac{1}{2}at^2$ ব্যবহার করবে।

ফিরিস্তি ৫.১৩: গতির সমীকরণ সমাধান (Solving Motion Equations)

```
float u, a, t;

cout << "adibeg toron somoy: ";
cin >> u >> a >> t;

float v = u + a * t;
float s = u*t + a * t * t / 2;

cout << "beg: " << v << " ";
cout << "durutto: " << s << endl;
```

যোগান-ফলনাংশ (input-output segment)

```
adibeg toron somoy: 2 1 4
beg: 6 durotto: 16
```

১০. নীচের ছদ্ম-সংকেতের (pseudocode) জন্য একটি ক্রমলেখ (program) তৈরী করো।

ক) পড়ো (read) x ও y
খ) গণো (compute) $p = x * y$
গ) গণো (compute) $s = x + y$
ঘ) গণো (compute) $t = s^2 + p * (s - x) * (p + y)$
ঙ) লিখো (write) t

ফিরিস্তি ৫.১৪: ছদ্মসংকেত থেকে ক্রমলেখ তৈরী (Program from Pseudocode)

```
int x, y;   // কেবল main বিপাতকের ভিতরের অংশটুকু

cin >> x >> y;  // ধাপ ক
int p = x * y;  // ধাপ খ
int s = x + y;  // ধাপ গ
int t = s*s + p * (s - x) * (p + y);  // ধাপ ঘ

cout << t << endl;  // ধাপ ঙ
```

৫.১২ গণনা পরিভাষা (Computing Terminologies)

- ধনাত্মক (positive)
- ঋণাত্মক (negative)
- অণুক্রিয়া (operator)
- উপাদান (operand)
- একিক (unary)
- দুয়িক (binary)
- উপাত্ত প্রকারান্তর (type casting)
- চলা-কালীন (run-time)
- নির্বাহ-কালীন (execution-time)
- টীকা দেয়া (commenting)
- টীকা তোলা (uncommenting)
- যোজন (composition)
- নর্থক (void)
- পৃথকী (separator)
- অগ্রগণ্যতা (precedence)
- সহযোজ্যতা (associativity)