C Programming 1st Note

ব্রাঞ্চিং স্টেটমেন্ট: সি প্রোগ্রামিং এবং বাস্তব জীবনের উদাহরণ

ব্রাঞ্চিং স্টেটমেন্ট হলো প্রোগ্রামিং লজিকের একটি গুরুত্বপূর্ণ অংশ, যা প্রোগ্রামের কার্যপ্রবাহকে বিভিন্ন পথে পরিচালিত করে। সি প্রোগ্রামিং ভাষায় ব্রাঞ্চিং স্টেটমেন্ট ব্যবহার করে নির্দিষ্ট শর্ত অনুযায়ী বিভিন্ন সিদ্ধান্ত নেওয়া এবং বিভিন্ন কোড ব্লক কার্যকর করা যায়। এই প্রবন্ধে আমরা ব্রাঞ্চিং স্টেটমেন্টের প্রকারভেদ এবং তাদের বাস্তব জীবনের ব্যবহার সম্পর্কে বিস্তারিত আলোচনা করব।

ব্রাঞ্চিং স্টেটমেন্ট কী?

ব্রাঞ্চিং স্টেটমেন্ট এমন একটি টুল, যা প্রোগ্রামকে শর্ত অনুযায়ী ভিন্ন ভিন্ন নির্দেশ (instruction) কার্যকর করতে সাহায্য করে। সহজ কথায়, ব্রাঞ্চিং স্টেটমেন্ট ব্যবহার করে আমরা প্রোগ্রামকে বিভিন্ন পথে নিয়ে যেতে পারি।

সি প্রোগ্রামিং-এ ব্রাঞ্চিং স্টেটমেন্টের প্রকারভেদ

সি ভাষায় ব্রাঞ্চিং স্টেটমেন্ট প্রধানত তিন ধরনের হয়:

1. **if স্টেটমেন্ট**
2. **switch স্টেটমেন্ট**
3. **goto স্টেটমেন্ট**

---

১. if স্টেটমেন্ট

`if` স্টেটমেন্ট শর্ত যাচাই করার জন্য ব্যবহৃত হয়। শর্ত সত্য (true) হলে নির্দিষ্ট কোড ব্লক কার্যকর হয়, আর মিথ্যা (false) হলে এটি এড়িয়ে যায়।

ধরন:

- **if**

- **if-else**
- **if-else if-else**

Syntax:

```
if (condition) {
    // শর্ত সত্য হলে এই কোড কার্যকর হবে
} else {
    // শর্ত মিথ্যা হলে এই কোড কার্যকর হবে
}
```

বাস্তব জীবনের উদাহরণ: ছাড় গণনা

একটি দোকানে যদি ১০০০ টাকার বেশি কেনাকাটা হয়, তাহলে গ্রাহক ১০% ছাড় পাবে।

```
#include <stdio.h>
int main() {
    int amount;
    printf("আপনার মোট কেনাকাটার পরিমাণ লিখুন: ");
    scanf("%d", &amount);

    if (amount > 1000) {
        printf("আপনি ১০ শতাংশ ছাড় পেয়েছেন।\n");
    } else {
        printf("আপনার কোনো ছাড় নেই।\n");
    }

    return 0;
}
```

ব্যবহারক্ষেত্র: ডিসকাউন্ট গণনা, বৈধতা যাচাই (validation)।

---

২. switch স্টেটমেন্ট

`switch` স্টেটমেন্ট তখন ব্যবহার করা হয় যখন একাধিক শর্ত যাচাই করতে হয়। এটি বিভিন্ন কেস অনুযায়ী নির্ধারিত কোড কার্যকর করে।

Syntax:

```
switch (expression) {
    case value1:
        // value1 এর জন্য কোড
        break;
    case value2:
        // value2 এর জন্য কোড
        break;
    default:
        // কোনো কেস না মিললে এই কোড কার্যকর হবে
}
```

বাস্তব জীবনের উদাহরণ: মেনু ভিত্তিক প্রোগ্রাম

ধরা যাক, একজন ব্যবহারকারী বিভিন্ন অপশন থেকে একটি নির্বাচন করবে।

```
#include <stdio.h>
int main() {
    int choice;
    printf("মেনু:\n");
    printf("1. চা\n2. কফি\n3. পানি\n");
    printf("আপনার পছন্দ দিন (1-3): ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("আপনার চা পরিবেশন করা হচ্ছে।\n");
            break;
        case 2:
            printf("আপনার কফি পরিবেশন করা হচ্ছে।\n");
            break;
        case 3:
            printf("আপনার পানি পরিবেশন করা হচ্ছে।\n");
            break;
        default:
```

```
            printf("ভুল পছন্দ। মেনু থেকে সঠিক সংখ্যা দিন।\n");
    }

    return 0;
}
```

ব্যবহারক্ষেত্র: রেস্টুরেন্ট মেনু, ইউজার ইন্টারফেস।

---

৩. goto স্টেটমেন্ট

`goto` স্টেটমেন্ট সরাসরি প্রোগ্রামের অন্য একটি অংশে লাফ দেওয়ার জন্য ব্যবহৃত হয়। এটি ব্যবহার করা নিরুৎসাহিত করা হয় কারণ এটি প্রোগ্রামের গঠনশীলতা নষ্ট করতে পারে। তবে কিছু নির্দিষ্ট পরিস্থিতিতে এটি উপকারী।

Syntax:

```
goto label;
// অন্যান্য কোড
label:
// লেবেলে সংযুক্ত কোড
```

বাস্তব জীবনের উদাহরণ: ত্রুটি পরিচালনা (Error Handling)

```
#include <stdio.h>
int main() {
    int number;
    printf("একটি পজিটিভ সংখ্যা দিন: ");
    scanf("%d", &number);

    if (number < 0) {
        goto error;
    }

    printf("আপনার দেওয়া সংখ্যা: %d\n", number);
    return 0;
```

```
error:
    printf("ত্রুটি: পজিটিভ সংখ্যা দিন।\n");
    return 1;
}
```

ব্যবহারক্ষেত্র: ত্রুটি পরিচালনা, লজিকাল জাম্প।

বাস্তব জীবনের আরও ব্যবহার

১. গ্রেড গণনা

শিক্ষার্থীর নম্বর অনুযায়ী গ্রেড নির্ধারণ করতে ব্রাঞ্চিং স্টেটমেন্ট ব্যবহার করা যায়।

```
#include <stdio.h>
int main() {
    int marks;
    printf("আপনার প্রাপ্ত নম্বর লিখুন: ");
    scanf("%d", &marks);

    if (marks >= 90) {
        printf("আপনার গ্রেড: A\n");
    } else if (marks >= 80) {
        printf("আপনার গ্রেড: B\n");
    } else if (marks >= 70) {
        printf("আপনার গ্রেড: C\n");
    } else {
        printf("আপনার গ্রেড: F\n");
    }

    return 0;
}
```

ব্যবহারক্ষেত্র: একাডেমিক গ্রেডিং সিস্টেম।

২. ব্যাংক অ্যাকাউন্ট লেনদেন

ব্যবহারকারী ডিপোজিট বা উত্তোলনের অপশন বেছে নিতে পারে।

```
#include <stdio.h>
int main() {
    int choice;
    float balance = 1000.0, amount;

    printf("ব্যাংক মেনু:\n");
    printf("1. টাকা জমা\n2. টাকা উত্তোলন\n3. ব্যালেন্স দেখুন\n");
    printf("আপনার পছন্দ দিন: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("জমার পরিমাণ লিখুন: ");
            scanf("%f", &amount);
            balance += amount;
            printf("নতুন ব্যালেন্স: %.2f\n", balance);
            break;
        case 2:
            printf("উত্তোলনের পরিমাণ লিখুন: ");
            scanf("%f", &amount);
            if (amount > balance) {
                printf("পর্যাপ্ত ব্যালেন্স নেই।\n");
            } else {
                balance -= amount;
                printf("উত্তোলনের পর সফল ব্যালেন্স: %.2f\n", balan
ce);
            }
            break;
        case 3:
            printf("বর্তমান ব্যালেন্স: %.2f\n", balance);
            break;
        default:
            printf("ভুল অপশন। সঠিক অপশন দিন।\n");
    }
```

```
    return 0;
}
```

ব্যবহারক্ষেত্র: ব্যাঙ্কিং সফটওয়্যার।

সি প্রোগ্রামিং-এ লুপিং স্টেটমেন্ট: একটি বিশদ আলোচনা

সি প্রোগ্রামিং ভাষায় **লুপিং স্টেটমেন্ট** হলো প্রোগ্রামিং লজিকের একটি গুরুত্বপূর্ণ অংশ। এটি প্রোগ্রামের একটি নির্দিষ্ট অংশ বারবার পুনরাবৃত্তি করার জন্য ব্যবহৃত হয়। লুপিং কৌশল ব্যবহার করে আমরা সময় বাঁচাতে পারি এবং কোডকে ছোট ও কার্যকর করতে পারি। এই প্রবন্ধে আমরা সি প্রোগ্রামিং-এ ব্যবহৃত বিভিন্ন লুপিং স্টেটমেন্ট যেমন **for loop**, **while loop**, এবং **do-while loop** সম্পর্কে বিশদে আলোচনা করব।

লুপের ধারণা

লুপ হলো এমন একটি কৌশল যেখানে একই কোড বারবার পুনরায় চালানো হয় যতক্ষণ পর্যন্ত নির্দিষ্ট শর্ত পূরণ হয়। লুপের প্রধান উপাদানগুলো হলো:

1. **শুরু করার অবস্থান (Initialization):** লুপ শুরু করার জন্য একটি প্রাথমিক মান সেট করা হয়।
2. **শর্ত (Condition):** একটি নির্দিষ্ট শর্ত যা লুপের কার্যক্রম চালিয়ে যাওয়ার জন্য পর্যালোচনা করা হয়।
3. **পরিবর্তন/হালনাগাদ (Update):** প্রতি পুনরাবৃত্তির পর শর্ত পূরণ না হওয়া পর্যন্ত মান আপডেট করা হয়।

সি প্রোগ্রামিং-এর প্রধান লুপিং স্টেটমেন্ট

সি ভাষায় প্রধানত তিন প্রকারের লুপ ব্যবহৃত হয়:

1. **for loop**
2. **while loop**
3. **do-while loop**

১. for loop

`for loop` হলো সবচেয়ে বেশি ব্যবহৃত লুপ। এটি তখন ব্যবহৃত হয় যখন পুনরাবৃত্তির সংখ্যা পূর্বে থেকেই নির্ধারিত থাকে।

Syntax:

```
for (initialization; condition; update) {
    // লুপের মধ্যে চালানোর কোড
}
```

উদাহরণ:

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

আউটপুট:

```
নম্বর: 1
নম্বর: 2
নম্বর: 3
নম্বর: 4
নম্বর: 5
```

ব্যাখ্যা:

1. **Initialization:** `i = 1` দিয়ে লুপ শুরু হয়।
2. **Condition:** `i <= 5` শর্তটি লুপ চালানোর অনুমতি দেয়।
3. **Update:** প্রতি পুনরাবৃত্তির শেষে `i++` দ্বারা মান এক বৃদ্ধি পায়।

---

২. while loop

`while loop` এমন ক্ষেত্রে ব্যবহার করা হয় যখন পুনরাবৃত্তি চালানোর শর্ত পূর্বেই জানা থাকে।

Syntax:

```
while (condition) {
    // লুপের মধ্যে চালানোর কোড
}
```

উদাহরণ:

```
#include <stdio.h>
int main() {
    int i = 1;
    while (i <= 5) {
        printf("নম্বর: %d\n", i);
        i++;
    }
    return 0;
}
```

আউটপুট:

```
নম্বর: 1
নম্বর: 2
নম্বর: 3
নম্বর: 4
নম্বর: 5
```

ব্যাখ্যা:

1. **Condition:** লুপটি চালু হওয়ার আগে `i <= 5` শর্তটি পরীক্ষা করা হয়।
2. **Update:** `i++` দ্বারা প্রতিটি পুনরাবৃত্তির পর মান এক বৃদ্ধি পায়।

---

৩. do-while loop

`do-while loop` হলো বিশেষ একটি লুপ, যেখানে কমপক্ষে একবার লুপের কোড অবশ্যই চালানো হয়। এর শর্তটি পুনরাবৃত্তির শেষে পরীক্ষা করা হয়।

Syntax:

```
do {
    // লুপের মধ্যে চালানোর কোড
} while (condition);
```

উদাহরণ:

```
#include <stdio.h>
int main() {
    int i = 1;
    do {
        printf("নম্বর: %d\n", i);
        i++;
    } while (i <= 5);
    return 0;
}
```

আউটপুট:

```
নম্বর: 1
নম্বর: 2
নম্বর: 3
নম্বর: 4
নম্বর: 5
```

ব্যাখ্যা:

1. প্রথমে লুপের কোড চালানো হয়।
2. পরে শর্তটি (`i <= 5`) পরীক্ষা করা হয়।
3. শর্ত সত্য হলে লুপ পুনরায় চালানো হয়।

লুপ ব্রেক এবং কন্টিনিউ স্টেটমেন্ট

break স্টেটমেন্ট

`break` স্টেটমেন্ট লুপ থেকে নির্ধারিত শর্ত পূরণ হলে সম্পূর্ণ লুপ বন্ধ করে দেয়।

উদাহরণ:

```
#include <stdio.h>
int main() {
    int i;
    for (i = 1; i <= 5; i++) {
        if (i == 3) {
            break;
        }
        printf("নম্বর: %d\n", i);
    }
    return 0;
}
```

আউটপুট:

```
নম্বর: 1
নম্বর: 2
```

continue স্টেটমেন্ট

`continue` স্টেটমেন্ট লুপের বাকি অংশ এড়িয়ে পরবর্তী পুনরাবৃত্তি শুরু করে।

উদাহরণ:

```
#include <stdio.h>
int main() {
    int i;
    for (i = 1; i <= 5; i++) {
        if (i == 3) {
            continue;
        }
```

```
        printf("নম্বর: %d\n", i);
    }
    return 0;
}
```

আউটপুট:

```
নম্বর: 1
নম্বর: 2
নম্বর: 4
নম্বর: 5
```

---

for loop, while loop, এবং do-while loop এর মধ্যে পার্থক্য

সি প্রোগ্রামিংয়ের তিন ধরনের লুপ— **for loop**, **while loop**, এবং **do-while loop**—একই উদ্দেশ্যে ব্যবহার করা হলেও, এদের কাঠামো, কার্যপ্রণালী, এবং ব্যবহারক্ষেত্রে ভিন্নতা রয়েছে। নিচে এদের মধ্যে মূল পার্থক্যগুলি বর্ণনা করা হলো:

পার্থক্যের দিক	for loop	while loop	do-while loop
শুরু করার প্রক্রিয়া	লুপের শিরোনামেই (header) প্রাথমিক মান নির্ধারণ করা হয়।	লুপের শিরোনামের বাইরে প্রাথমিক মান নির্ধারণ করতে হয়।	লুপের শিরোনামের বাইরে প্রাথমিক মান নির্ধারণ করতে হয়।
শর্ত যাচাই	লুপ চালানোর আগে শর্ত যাচাই করা হয়।	লুপ চালানোর আগে শর্ত যাচাই করা হয়।	লুপের কোড একবার চালানোর পর শর্ত যাচাই করা হয়।
চালানোর নিশ্চয়তা	শর্ত প্রথমে সত্য হলে তবেই লুপ চালানো হয়।	শর্ত প্রথমে সত্য হলে তবেই লুপ চালানো হয়।	লুপ কমপক্ষে একবার অবশ্যই চালানো হয়, শর্ত মিথ্যা হলেও।
স্ট্রাকচার	এক লাইনে শুরু, শর্ত এবং আপডেট উল্লেখ করা হয়।	শর্ত এবং আপডেট পৃথকভাবে উল্লেখ করতে হয়।	লুপের কোড চালানোর পর শর্ত উল্লেখ করা হয়।
ব্যবহারক্ষেত্র	পুনরাবৃত্তির সংখ্যা আগে থেকে জানা থাকলে ব্যবহার করা হয়।	পুনরাবৃত্তি চালানোর শর্ত আগে থেকে জানা থাকলে ব্যবহার করা হয়।	যখন শর্ত যাচাইয়ের আগে লুপ একবার চালানো আবশ্যক।

উদাহরণ সহ ব্যাখ্যা

for loop

`for loop` পুনরাবৃত্তির সংখ্যা আগে থেকেই নির্ধারিত থাকলে ব্যবহার করা হয়।

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

আউটপুট:

```
নম্বর: 1
নম্বর: 2
নম্বর: 3
নম্বর: 4
নম্বর: 5
```

while loop

`while loop` তখন ব্যবহার করা হয় যখন শর্ত পূরণ না হওয়া পর্যন্ত লুপ চালানো প্রয়োজন।

```
#include <stdio.h>
int main() {
    int i = 1;
    while (i <= 5) {
        printf("নম্বর: %d\n", i);
        i++;
    }
    return 0;
}
```

আউটপুট:

```
নম্বর: 1
নম্বর: 2
নম্বর: 3
নম্বর: 4
নম্বর: 5
```

do-while loop

`do-while loop` ব্যবহার করা হয় যখন লুপ একবার চালানো আবশ্যক, তারপর শর্ত যাচাই হয়।

```c
#include <stdio.h>
int main() {
    int i = 1;
    do {
        printf("নম্বর: %d\n", i);
        i++;
    } while (i <= 5);
    return 0;
}
```

আউটপুট:

```
নম্বর: 1
নম্বর: 2
নম্বর: 3
নম্বর: 4
নম্বর: 5
```

সংক্ষিপ্ত তুলনা

লুপের ধরন	শর্ত আগে যাচাই হয়?	লুপ অন্তত একবার চালানো হয়?	ব্যবহারক্ষেত্র
for loop	হ্যাঁ	না	পুনরাবৃত্তির সংখ্যা নির্ধারিত থাকলে।

while loop	হ্যাঁ	না	শর্ত পূরণ না হওয়া পর্যন্ত চালাতে।
do-while loop	না	হ্যাঁ	শর্ত যাচাইয়ের আগে একবার চালাতে।

লুপের বাস্তব জীবনের ব্যবহার: সি প্রোগ্রামিং-এ উদাহরণ সহ

লুপ হল এমন একটি টুল যা প্রোগ্রামিংয়ে পুনরাবৃত্তিমূলক কাজ সহজ করতে ব্যবহার করা হয়। বাস্তব জীবনের বিভিন্ন সমস্যার সমাধানে লুপ অত্যন্ত কার্যকর। নিচে লুপের কিছু বাস্তব জীবনের ব্যবহারিক উদাহরণ তুলে ধরা হলো:

১. নাম্বারের তালিকা তৈরি এবং প্রিন্ট করা

ধরা যাক, আপনাকে ১ থেকে ১০০ পর্যন্ত নাম্বার প্রিন্ট করতে বলা হয়েছে। এটি ম্যানুয়ালি করলে অনেক সময় লাগবে। কিন্তু লুপ ব্যবহার করে সহজেই কাজটি করা সম্ভব।

উদাহরণ:

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

ব্যবহারক্ষেত্র: পরীক্ষার ফলাফলের তালিকা তৈরি, সিরিয়াল নম্বর তৈরি ইত্যাদি।

২. যোগফল নির্ণয় (Summation Calculation)

কোনো নির্দিষ্ট সংখ্যার পরিসরের যোগফল বের করার জন্য লুপ ব্যবহার করা হয়।

উদাহরণ:
১ থেকে ১০ পর্যন্ত সংখ্যাগুলোর যোগফল বের করা:

```c
#include <stdio.h>
int main() {
    int sum = 0;
    for (int i = 1; i <= 10; i++) {
        sum += i;
    }
    printf("১ থেকে ১০ পর্যন্ত সংখ্যার যোগফল: %d\n", sum);
    return 0;
}
```

ব্যবহারক্ষেত্র: হিসাব-নিকাশে যেমন মাসিক ব্যয়ের যোগফল বা বিলিং সিস্টেম।

---

৩. টেবিল বা মাল্টিপ্লিকেশন টেবিল তৈরি

লুপ ব্যবহার করে খুব সহজে একটি সংখ্যার গুণের টেবিল তৈরি করা যায়।

উদাহরণ:
ধরা যাক, ৫-এর গুণের টেবিল প্রিন্ট করতে হবে:

```c
#include <stdio.h>
int main() {
    int num = 5;
    for (int i = 1; i <= 10; i++) {
        printf("%d x %d = %d\n", num, i, num * i);
    }
    return 0;
}
```

ব্যবহারক্ষেত্র: শিক্ষা বা গণনামূলক কাজে।

---

৪. ব্যবহারকারীর ইনপুট প্রক্রিয়াকরণ

লুপ ব্যবহার করে ব্যবহারকারীর একাধিক ইনপুট নেওয়া এবং প্রক্রিয়াকরণ করা যায়।

উদাহরণ:
৫টি সংখ্যার যোগফল নির্ণয়:

```c
#include <stdio.h>
int main() {
    int sum = 0, num;
    for (int i = 1; i <= 5; i++) {
        printf("সংখ্যা দিন (%d): ", i);
        scanf("%d", &num);
        sum += num;
    }
    printf("সংখ্যাগুলোর মোট যোগফল: %d\n", sum);
    return 0;
}
```

ব্যবহারক্ষেত্র: ফর্ম ডেটা প্রসেসিং বা রিপোর্ট জেনারেশনের সময়।

---

৫. ডেটা অনুসন্ধান বা প্রাপ্তি (Data Search)

একটি লিস্ট বা অ্যারে থেকে নির্দিষ্ট তথ্য খুঁজে বের করার জন্য লুপ ব্যবহার করা হয়।

উদাহরণ:
অ্যারে থেকে ৫ সংখ্যাটি খুঁজে পাওয়া:

```c
#include <stdio.h>
int main() {
    int arr[] = {3, 8, 5, 12, 9};
    int size = 5, found = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] == 5) {
            found = 1;
            printf("৫ নাম্বারটি %d তম স্থানে পাওয়া গেছে।\n", i + 1);
            break;
        }
    }
    if (!found) {
        printf("৫ নাম্বারটি পাওয়া যায়নি।\n");
    }
```

```
    return 0;
}
```

ব্যবহারক্ষেত্র: ডেটাবেস অনুসন্ধান বা ফাইল সিস্টেমে ডেটা খুঁজে বের করা।

---

৬. ফ্যাক্টরিয়াল হিসাব করা

ফ্যাক্টরিয়াল (n!) নির্ণয়ের জন্য লুপ খুবই কার্যকর।

উদাহরণ:
৫! (৫ ফ্যাক্টরিয়াল) নির্ণয়:

```
#include <stdio.h>
int main() {
    int n = 5, fact = 1;
    for (int i = 1; i <= n; i++) {
        fact *= i;
    }
    printf("৫! (ফ্যাক্টরিয়াল) হলো: %d\n", fact);
    return 0;
}
```

ব্যবহারক্ষেত্র: গণিত, পরিসংখ্যান এবং কম্পিউটার সায়েন্সের সমস্যা সমাধান।

---

৭. গ্রাফিক্স বা প্যাটার্ন আঁকা

লুপ ব্যবহার করে বিভিন্ন প্যাটার্ন তৈরি করা যায় যা গ্রাফিক্স বা অ্যাস্কি আর্ট তৈরিতে ব্যবহৃত হয়।

উদাহরণ:
তিন কোণের প্যাটার্ন আঁকা:

```
#include <stdio.h>
int main() {
    int rows = 5;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("* ");
        }
```

```
        printf("\n");
    }
    return 0;
}
```

আউটপুট:

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

ব্যবহারক্ষেত্র: ইউজার ইন্টারফেস ডিজাইন বা গ্রাফিক প্রোগ্রামিং।

---

৮. সার্চ ইঞ্জিন লজিক বা ফিল্টারিং

একটি লিস্ট থেকে নির্দিষ্ট শর্তে ডেটা বের করার ক্ষেত্রে লুপ ব্যবহৃত হয়। যেমন: একটি ই-কমার্স সাইটে ৫০০ টাকার বেশি দামের প্রোডাক্ট ফিল্টার করা।

---

do-while loop এর বাস্তব জীবনের ব্যবহার: উদাহরণসহ বিশ্লেষণ

do-while loop হলো এমন একটি লুপ যা কমপক্ষে একবার চালানো হয়, কারণ শর্ত যাচাই করার আগে লুপের কোড চালানো হয়। এটি এমন ক্ষেত্রে কার্যকর যেখানে কোনো কাজ অবশ্যই একবার সম্পাদন করতে হবে এবং তারপর শর্ত অনুযায়ী পুনরাবৃত্তি করতে হবে।

নিচে বাস্তব জীবনের বিভিন্ন উদাহরণে **do-while loop** এর ব্যবহার ব্যাখ্যা করা হলো:

---

১. পুনরায় ব্যবহারকারীর ইনপুট চাওয়া

যখন ব্যবহারকারীকে সঠিক ইনপুট না দেওয়া পর্যন্ত বারবার ইনপুট চাওয়া হয়, তখন **do-while loop** ব্যবহার করা যায়।

উদাহরণস্বরূপ, পিন বা পাসওয়ার্ড যাচাই করার ক্ষেত্রে:

```
#include <stdio.h>
int main() {
    int password;
```

```
    do {
        printf("পাসওয়ার্ড দিন (সঠিক পাসওয়ার্ড: 1234): ");
        scanf("%d", &password);
    } while (password != 1234);
    printf("সঠিক পাসওয়ার্ড দেওয়া হয়েছে।\n");
    return 0;
}
```

ব্যবহারক্ষেত্র: এটিএম মেশিনে পিন যাচাই, লগইন সিস্টেম।

---

২. মেনু-ড্রাইভেন প্রোগ্রাম

কোনো মেনু থেকে ব্যবহারকারীকে অপশন বেছে নিতে দেওয়া এবং পুনরায় মেনু দেখানো পর্যন্ত **do-while loop** ব্যবহার করা হয়।

উদাহরণ:

```
#include <stdio.h>
int main() {
    int choice;
    do {
        printf("\nমেনু:\n");
        printf("1. যোগফল দেখুন\n");
        printf("2. বিয়োগফল দেখুন\n");
        printf("3. প্রস্থান করুন\n");
        printf("আপনার পছন্দ দিন: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("যোগফল: 10 + 5 = 15\n");
                break;
            case 2:
                printf("বিয়োগফল: 10 - 5 = 5\n");
                break;
            case 3:
                printf("প্রোগ্রাম বন্ধ করা হচ্ছে।\n");
```

```
                break;
            default:
                printf("সঠিক পছন্দ দিন।\n");
        }
    } while (choice != 3);
    return 0;
}
```

ব্যবহারক্ষেত্র:

- সফটওয়্যারে মেনু ভিত্তিক ইন্টারফেস।
- ব্যাংকিং বা রেস্টুরেন্ট ম্যানেজমেন্ট সিস্টেম।

---

৩. ডেটা প্রসেসিং (অন্তত একবার প্রক্রিয়া করা)

ধরা যাক, ব্যবহারকারীর একাধিক নাম প্রক্রিয়া করতে হবে, তবে প্রথমে অন্তত একটি নাম ইনপুট নিতে হবে।

```
#include <stdio.h>
int main() {
    char name[50];
    char repeat;
    do {
        printf("আপনার নাম লিখুন: ");
        scanf("%s", name);
        printf("স্বাগতম, %s!\n", name);

        printf("আরও নাম দিতে চান? (y/n): ");
        scanf(" %c", &repeat);
    } while (repeat == 'y');
    return 0;
}
```

ব্যবহারক্ষেত্র:

- ব্যবহারকারীর তথ্য সংগ্রহ।

- ফর্ম ইনপুটের প্রক্রিয়াকরণ।

---

৪. গেমিং সিস্টেম

ধরা যাক, গেম খেলার সময় ব্যবহারকারীকে বারবার জিজ্ঞাসা করা হবে যে তারা আবার খেলতে চান কিনা।

```
#include <stdio.h>
int main() {
    char playAgain;
    do {
        printf("গেম চলছে...\n");
        printf("আপনি আবার খেলতে চান? (y/n): ");
        scanf(" %c", &playAgain);
    } while (playAgain == 'y');
    printf("গেম শেষ।\n");
    return 0;
}
```

ব্যবহারক্ষেত্র:

ভিডিও গেম বা বোর্ড গেম সিস্টেম যেখানে ব্যবহারকারী বারবার খেলার সুযোগ পান।

---

৫. অনুমানমূলক গেম (Guessing Game)

ব্যবহারকারীকে একটি সঠিক সংখ্যা অনুমান করতে বলা হয়। সঠিক উত্তর দেওয়ার আগ পর্যন্ত এটি চলবে।

```
#include <stdio.h>
int main() {
    int number = 7, guess;
    do {
        printf("একটি সংখ্যা অনুমান করুন (১-১০): ");
        scanf("%d", &guess);
        if (guess != number) {
            printf("ভুল অনুমান! আবার চেষ্টা করুন।\n");
        }
```

```
    } while (guess != number);
    printf("অভিনন্দন! আপনি সঠিক সংখ্যা অনুমান করেছেন।\n");
    return 0;
}
```

ব্যবহারক্ষেত্র:

- গেম ডিজাইন।
- শিক্ষা বা মজার কার্যক্রম।

---

৬. ফাইল পড়া এবং প্রক্রিয়াকরণ

ফাইলের ডেটা পড়তে এবং শেষ পর্যন্ত প্রক্রিয়া চালাতে **do-while loop** ব্যবহার করা হয়।

ধরা যাক, একটি ফাইলের প্রতিটি লাইন পড়া হচ্ছে।

```
#include <stdio.h>
int main() {
    FILE *file;
    char line[100];
    file = fopen("data.txt", "r");
    if (file == NULL) {
        printf("ফাইল খোলা যায়নি।\n");
        return 1;
    }

    do {
        if (fgets(line, sizeof(line), file)) {
            printf("%s", line);
        }
    } while (!feof(file));

    fclose(file);
    return 0;
}
```

ব্যবহারক্ষেত্র:

- ফাইল প্রসেসিং সিস্টেম।
- ডেটা এনালাইসিস বা লগ পড়া।
