মডিউল ১৭ঃ Recursion

Previous
মডিউল ১৫-৭ঃ ফাংশন ইউথ এরে এস রেফারেন্স

Next
মডিউল ১৭-১ঃ Call Stack

মডিউল ১৭-১ঃ Call Stack

কল স্ট্যাক" (Call Stack) হলো কম্পিউটার প্রোগ্রামিং এর একটি গুরুত্বপূর্ণ ধারণা, বিশেষ করে ফাংশন কল (function call) এর ক্ষেত্রে। C ভাষাতেও Call Stack একটি গুরুত্বপূর্ণ ভূমিকা রাখে।

কল স্ট্যাক কি?

- কল স্ট্যাক হলো LIFO (Last In First Out) নীতি ভিত্তিক একটি ডাটা স্ট্রাকচার। এটি মূলত ফাংশন কল এর History ট্র্যাক করে রাখে।
- প্রতিটি ফাংশন কল এর সময় তা কল স্ট্যাকে এড করে দেওয়া হয় (pushed)।
- যখন কোন ফাংশন তার কাজ সম্পন্ন করে, তখন সেই ফাংশনের তথ্য কল স্ট্যাক থেকে মুছে ফেলা হয় (popped)।

কল স্ট্যাক C ভাষায় কিভাবে কাজ করে?

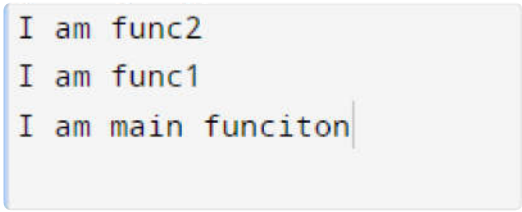
উদাহরণ:

```
#include<stdio.h>

void func2() {
  printf("I am func2\n");
}
void func1() {
  func2();
  printf("I am func1\n");
}
int main() {
  func1();
  printf("I am main funciton\n");
  return 0;
}
```

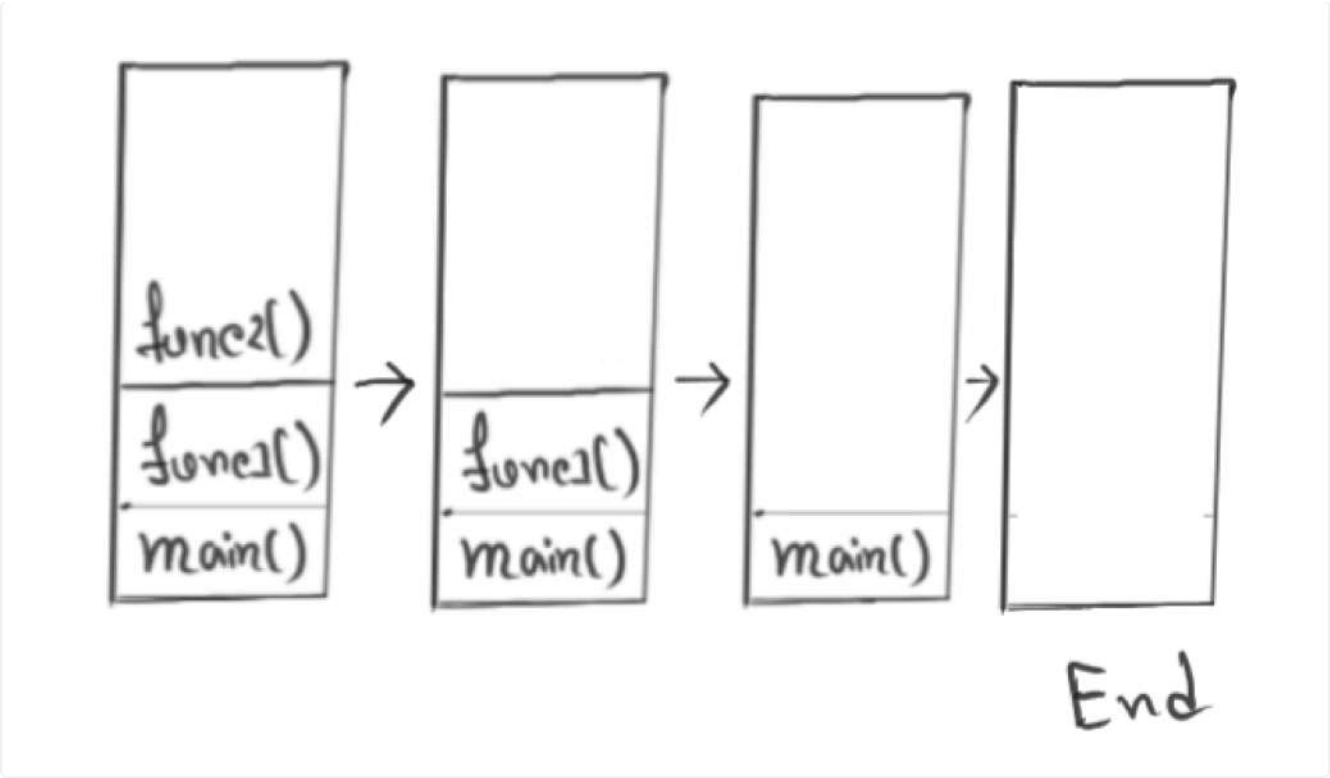
এই উদাহরণে, `main` ফাংশন `func1` কে কল করে। `func1` ফাংশনটি `func2` কে কল করে। কল স্ট্যাক এই ফাংশন কলগুলো ট্র্যাক করে রাখে ।

Output:



```
I am func2
I am func1
I am main funciton
```

Call Stack:



Previous
মডিউল ১৭ঃ Recursion

Next
মডিউল ১৭-২ঃ রিকার্সন

Last updated 5 months ago

মডিউল ১৭-২ঃ রিকার্সন

রিকার্সন (Recursion) কি?

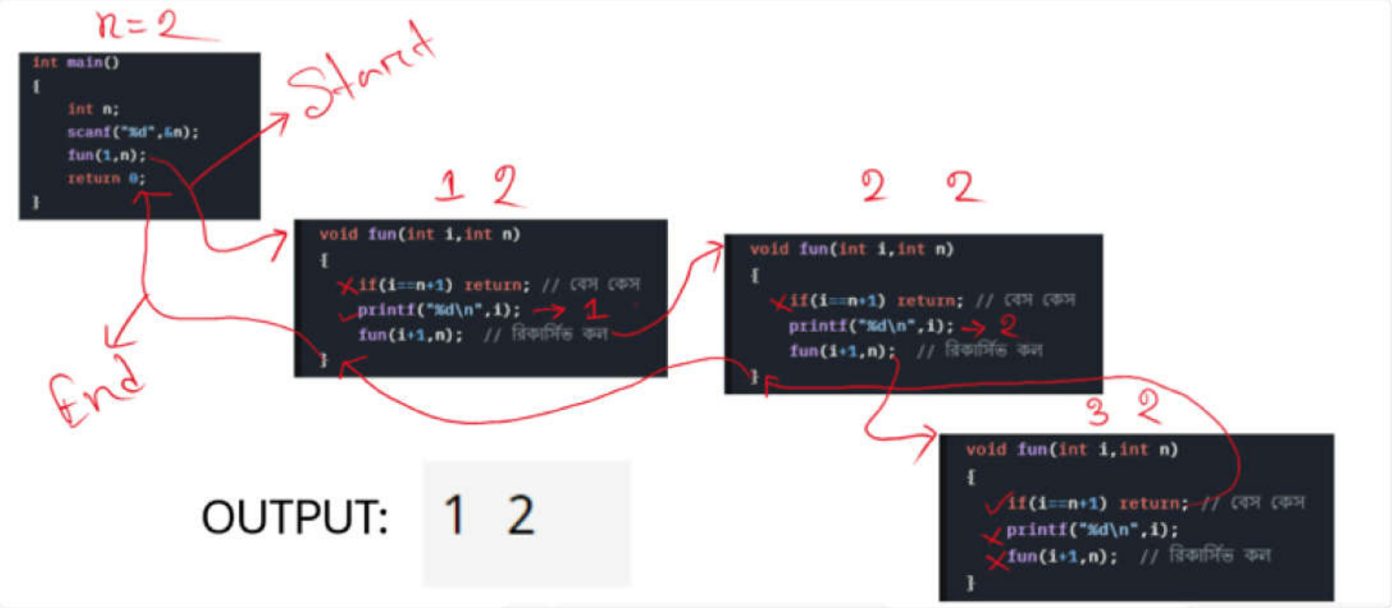
রিকার্সন হলো এমন একটি প্রোগ্রামিং কৌশল যেখানে একটি ফাংশন নিজেকেই কল করে। এটি সাধারণত সমস্যা সমাধানের জন্য ব্যবহৃত হয় যা সাব-প্রবলেম (sub-problem) থেকে গঠিত।

রিকার্সন কিভাবে কাজ করে:

1. **বেস কেস (Base Case):** রিকার্সিভ ফাংশনে এমন একটি শর্ত থাকতে হবে যা রিকার্সন বন্ধ করে দেয়। এই শর্তকে "বেস কেস" বলা হয়।
2. **রিকার্সিভ কল (Recursive Call):** রিকার্সিভ ফাংশন নিজেকেই নিজে কল করে, এবং কল করা ফাংশনটি মূল ফাংশনের সাব-প্রবলেম সমাধান করে।
3. **সমাধান একত্রিত করা (Combining Solutions):** রিকার্সিভ কল থেকে ফেরত আসার পর, মূল ফাংশন সাব-প্রবলেমের সমাধানগুলো একত্রিত করে সামগ্রিক সমাধান তৈরি করে।

উদাহরণ:

```c
#include<stdio.h>
void fun(int i,int n)
{
    if(i==n+1) return; // বেস কেস
    printf("%d\n",i);
    fun(i+1,n);  // রিকার্সিভ কল
}
int main()
{
    int n;
    scanf("%d",&n);
    fun(1,n);
    return 0;
}
```



```
int main()
{
    int n;
    scanf("%d",&n);
    fun(1,n);
    return 0;
}
```

```
void fun(int i,int n)
{
    if(i==n+1) return; // বেস কেস
    printf("%d\n",i);
    fun(i+1,n);  // রিকার্সিভ কল
}
```

```
void fun(int i,int n)
{
    if(i==n+1) return; // বেস কেস
    printf("%d\n",i);
    fun(i+1,n);  // রিকার্সিভ কল
}
```

```
void fun(int i,int n)
{
    if(i==n+1) return; // বেস কেস
    printf("%d\n",i);
    fun(i+1,n);  // রিকার্সিভ কল
}
```

OUTPUT:

```
1  2
```

Previous
মডিউল ১৭-১ঃ Call Stack

Next
মডিউল ১৭-৩ঃ Print From 1 to n Using Recursion

Last updated 5 months ago

মডিউল ১৭-৩ঃ Print From 1 to n Using Recursion

Code:

```
#include<stdio.h>
void fun(int i,int n)
{
    if(i==n+1) return; // Base case
    printf("%d\n",i);
    fun(i+1,n);
}
int main()
{
    int n;
    scanf("%d",&n);
    fun(1,n);
    return 0;
}
```

যদি n=3 হয়, তাহলে প্রোগ্রাম যেভাবে কাজ করবে তা নিচে দেখানো হলো:

1. `scanf("%d",&n);` দ্বারা প্রোগ্রাম n এর মান প্রাপ্ত করে। এখানে n=3 হলে, এটি একেবারে নিচের মতো হবে:

   ```
   n = 3
   ```

2. `fun(1,n);` কল হবে এবং সাথে দুটি আর্গুমেন্ট পাঠানো হবে i=1 এবং n=3।
3. `fun()` ফাংশনের মধ্যে, `i=1` এবং `n=3` হিসেবে প্যারামিটার পাস হয়েছে।
4. শুরুতে ফাংশনের ভেতরে যাওয়া হবে, `if(i==n+1) return;` শর্ত চেক করে, যে যদি `i` এর মান n+1 এর সমান হয়, তাহলে ফাংশন থেকে বের হয়ে যাওয়া হবে। এখানে i=1 এবং n=3, সুতরাং এই শর্ত পূরণ হবে না।
5. পরবর্তী লাইনে, `printf("%d\n",i);` দ্বারা i এর মান 1 প্রিন্ট করা হবে এবং নতুন লাইনে চলে আসবে। এই সময় আউটপুট হবে:

   ```
   1
   ```

6. পরবর্তী লাইনে, `fun(i+1,n);` দ্বারা ফাংশনকে নিজেকে নিজের মধ্যে কল করা হবে প্যারামিটার i এর মা এক বাড়িয়ে এবং এই সময় i=2 হবে।
7. আবার ফাংশনে প্রবেশ হবে, এবং এইবার প্রথম লাইনের শর্ত চেক হবে। যে প্রথম আগের প্রোগ্রামে এই শর্ত পূরণ হবে না।
8. এখন `printf("%d\n",i);` লাইনে i এর মান 2 প্রিন্ট হবে এবং নতুন লাইনে চলে আসবে। আউটপুট হবে:

```
1
2
```

9. আবার ফাংশনকে কল করে i এর মান 3 প্রিন্ট হবে এবং ফাংশন থেকে বের হয়ে আসা হবে কারণ i এর মান এখন n+1 এর সমান হয়েছে। আউটপুট হবে:

```
1
2
3
```

10. সবশেষে, মেইন ফাংশনে রিটার্ন ০ করে প্রোগ্রাম শেষ হবে।

Previous
মডিউল ১৭-২ঃ রিকার্সন

Next
মডিউল ১৭-৪ঃ Print From 5 to 1 using Recursion

Last updated 5 months ago

মডিউল ১৭-৪ঃ Print From 5 to 1 using Recursion

কোডঃ

```c
#include<stdio.h>
void fun(int i)
{
    // base case
    if(i==0) return;
    printf("%d\n",i);
    fun(i-1);
}
int main()
{
    fun(5);
    return 0;
}
```

এই প্রোগ্রামটি একটি সিম্পল রিকার্সিভ ফাংশন ব্যবহার করে একটি নির্দিষ্ট সংখ্যার পর্যন্ত পূর্ণসংখ্যাগুলি প্রিন্ট করে। এটির প্রোগ্রামের কাজ নিম্নলিখিত:

1. `fun()` ফাংশনটি নেওয়া হলো একটি পূর্ণসংখ্যা i এর মান নিয়ে।
2. যদি i এর মান 0 হয়, অর্থাৎ যদি i এর মান নিকটতম বেস কেস হয়, তবে ফাংশন থেকে বাহির হতে হবে। এই ধরণের কেস হলো বেস কেস বা মৌলিক কেস।
3. যদি i এর মান 0 না হয়, তবে একটি লাইন প্রিন্ট করা হবে `printf("%d\n",i);` এবং একটি রিকার্সিভ কল করা হবে `fun(i-1);` যেখানে i-1 হচ্ছে এই ফাংশনের আর্গুমেন্ট এবং এই রিকার্সিভ কল মাধ্যমে আমরা সংখ্যাগুলি ক্রমিকভাবে কমাতে থাকব।
4. প্রথমে ফাংশন কল হওয়ার সময়, সংখ্যা 5 প্রিন্ট করা হবে, এবং তারপরে সিস্টেম একটি রিকার্সিভ কল করে i-1=4 এর সাথে ফাংশনটি কল করবে।
5. পরের ধাপে, ফাংশন আবার কল হবে এবং সংখ্যা 4 প্রিন্ট করা হবে, এবং পরের সংখ্যা 3 এর জন্য আবার একটি রিকার্সিভ কল করা হবে।
6. এই পদক্ষেপ প্রক্রিয়া চলবে যতক্ষণ না i এর মান 0 হয়ে যায়। এই রিকার্সিভ কলের প্রতিটি সময় i এর মান 0 না হয়ে গেলে, তার আগের সংখ্যা প্রিন্ট হবে এবং সে সংখ্যার জন্য আবার একটি রিকার্সিভ কল করা হবে যার মান এক কম।

7. সবশেষে, যখন i এর মান 0 হয়ে যাবে, ফাংশন থেকে বের হতে হবে এবং মেইন ফাংশনে রিটার্ন ০ করে প্রোগ্রাম শেষ হবে।

এই রিকার্সিভ ফাংশনটি ব্যবহার করে আমরা একে একে সংখ্যাগুলি প্রিন্ট করে একে একে কমিয়ে যাচ্ছি। তার ফলে আমরা প্রথমে 5 থেকে শুরু করে 1 পর্যন্ত সংখ্যা প্রিন্ট করতে পারব।

Previous
মডিউল ১৭-৩ঃ Print From 1 to n Using Recursion

Next
মডিউল ১৭-৫ঃArray Printing using Recursion

Last updated 5 months ago

মডিউল ১৭-৫ঃArray Printing using Recursion

```c
#include<stdio.h>

void fun(int a[], int n, int i) {
    // বেস কেস: যদি i এর মান n এর সমান হয়
    if(i == n)
        return;

    // এই ইন্ডেক্সের মান প্রিন্ট করা
    printf("%d\n", a[i]);

    // পরবর্তী ইনডেক্সের জন্য রিকার্সিভ কল
    fun(a, n, i+1);
}

int main() {
    int n;
    scanf("%d", &n);
    int a[n];

    // অ্যারের উপাদানগুলি ইনপুট নেওয়া
    for(int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    // রিকার্সিভ ফাংশন কল
    fun(a, n, 0);

    return 0;
}
```

1. `fun()` ফাংশনটি নেওয়া হলো তিনটি আর্গুমেন্ট - একটি অ্যারে a[], অ্যারের মোট উপাদানের সংখ্যা n এবং একটি স্বাভাবিক সংখ্যা i।
2. যদি i এর মান n এর সমান হয়, অর্থাৎ যদি সার্বিক বেস কেস পূরণ হয়, তবে ফাংশন থেকে return হতে হবে।
3. আরেকটি, যদি বেস কেস পূরণ না হয়, তবে সেই সময়ে আমরা a[i] এর মান প্রিন্ট করব।
4. তারপর, আমরা আবার একটি রিকার্সিভ কল করব ফাংশনের মধ্যে তিনটি আর্গুমেন্ট পাস করে - অ্যারে, অ্যারের মোট উপাদানের সংখ্যা, এবং এখন কত ইনডেক্সে প্রিন্ট করা হয়েছে, এই ইনডেক্স এর মান এক বাড়িয়ে পরের উপাদান প্রিন্ট করতে যাব।
5. প্রথমে, মেইন ফাংশনে n এর মান ইনপুট হবে।

6. তারপর, মেইন ফাংশনের মধ্যে নির্দিষ্ট সংখ্যা প্রকারের উপাদান ধারার জন্য একটি অ্যারে তৈরি করা হবে এবং সেই উপাদানগুলি ইনপুট হবে।
7. সব তৈরি হওয়ার পরে, ফাংশন কল করা হবে সংখ্যা প্রিন্ট করার জন্য। এই ফাংশনে তিনটি প্যারামিটার পাস করা হবে - অ্যারে, অ্যারের মোট উপাদানের সংখ্যা, এবং প্রথম ইনডেক্স কে নির্দিষ্ট করার জন্য ইনডেক্স i।
8. আমরা একেবারে প্রথম ইনডেক্স থেকে শুরু করে শেষ ইনডেক্স পর্যন্ত প্রত্যেকটি উপাদান প্রিন্ট করব এবং পরবর্তী ইনডেক্স প্রিন্ট করার জন্য রিকার্সিভ কল করব।
9. প্রতিটি রিকার্সিভ কলে, i এর মান বাড়িয়ে পরের ইনডেক্স প্রিন্ট করার জন্য pass করা হবে।
10. প্রতিটি ইন্ডেক্সের মান প্রিন্ট করা সম্পন্ন হওয়ার পরে, প্রোগ্রাম সমাপ্তি পায়।

এই প্রোগ্রামটি ব্যবহার করে আমরা অ্যারের প্রতিটি উপাদানকে ক্রমানুসারে প্রিন্ট করতে পারি।

Previous
মডিউল ১৭-৪ঃ Print From 5 to 1 using Recursion

Next
মডিউল ১৭-৬ঃ Print From 5 to 1 in Reverse Way

Last updated 5 months ago

মডিউল ১৭-৬ঃ Print From 5 to 1 in Reverse Way

```c
#include<stdio.h>

void fun(int i) {
    // বেস কেস: যদি i এর মান 6 এর সমান হয়
    if(i == 6)
        return;

    // পরবর্তী সংখ্যার জন্য রিকার্সিভ কল
    fun(i + 1);

    // এই সময় প্রিন্ট করা হবে
    printf("%d\n", i);
}

int main() {
    // প্রথম সংখ্যা হিসেবে ১ পাস করানো
    fun(1);

    return 0;
}
```

ফাংশন fun() প্রথমে বেস কেস চেক করে। যদি i এর মান 6 এর সমান হয়, তবে ফাংশন থেকে বের হয়ে যাবে। এটি হচ্ছে বেস কেস বা মৌলিক কেস।

যদি বেস কেস পূরণ না হয়, তবে সেই সময়ে আমরা পরবর্তী সংখ্যার জন্য রিকার্সিভ কল করি। এই কলে আমরা i + 1 পাস করে দিচ্ছি, তারপরে সেই সংখ্যার জন্য প্রিন্ট করে দিচ্ছি।

যদি আমরা প্রোগ্রামটি চালাই, তবে আউটপুট হবে:

```
5
4
3
2
1
```

Previous

মডিউল ১৭-৫ঃArray Printing using Recursion

Next
মডিউল ১৭-৭ঃ Length of a String using Recursion

Last updated 5 months ago

মডিউল ১৭-৭ঃ Length of a String using Recursion

```
#include<stdio.h>

// রিকার্সিভ ফাংশন যা স্ট্রিংএর দৈর্ঘ্য হিসেব করে
int fun(char a[], int i) {
    // বেস কেস: যদি বর্তমান ইনডেক্সে নাল টার্মিনেটর পাওয়া যায়
    if(a[i] == '\0')
        return 0;

    // পরবর্তী ইনডেক্সের জন্য রিকার্সিভ কল করো
    int l = fun(a, i + 1);

    // এই সময় বর্তমান ইনডেক্সের মান + 1 রিটার্ন করো
    return l + 1;
}

int main() {
    char a[20] = "rahat";
    // রিকার্সিভ ফাংশনের কল করা
    int length = fun(a, 0);
    printf("%d\n", length);
    return 0;
}
```

এখানে, ফাংশন fun() প্রথমে বেস কেস চেক করে। যদি বর্তমান ইনডেক্সে নাল টার্মিনেটর পাওয়া যায়, তাহলে ফাংশন থেকে ০ রিটার্ন করে। এটি হচ্ছে বেস কেস বা মৌলিক কেস।

যদি বেস কেস পূরণ না হয়, তবে সেই সময়ে আমরা পরবর্তী ইনডেক্সের জন্য রিকার্সিভ কল করি। এই কলে আমরা i + 1 পাস করে দিচ্ছি, তারপরে সেই ইনডেক্সের মানের প্রত্যেকটির জন্য প্রিভিয়াস রিকার্সিভ কল করে আগের লেনথ পাওয়া হয়।

যদি আমরা প্রোগ্রামটি চালাই, তবে আউটপুট হবে:

```
5
```

স্ট্রিং "rahat" এর লেনথ হচ্ছে 5।

Previous

মডিউল ১৭-৬ঃ Print From 5 to 1 in Reverse Way

Next
মডিউল ১৮ঃ 2D এরে

Last updated 5 months ago