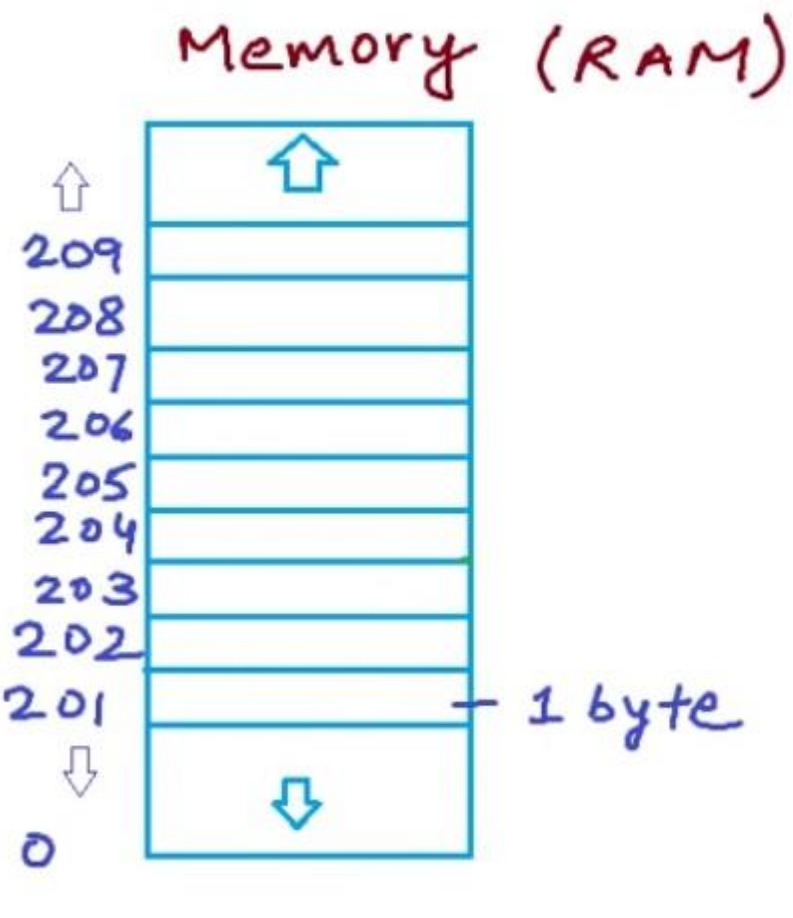
সি প্রোগ্রামিংঃ পয়েন্টার

কম্পিউটার মেমরি এবং মেমরি অ্যাড্রেস

পয়েন্টার প্রোগ্রামিং এ দারুন একটি টুল। পয়েন্টার সম্পর্কে জানার আগে কিছু ব্যাসিক জিনিস জানা যাক, যেগুলো বুঝতে কাজে দিবে।

ভ্যারিয়েবল গুলো কিভাবে কম্পিউটার মেমরিতে/ র‍্যাম এ স্টোর হয়?

র‍্যাম এর এক একটি সেল এক একটি বাইট। আর প্রত্যেকটা বাইট এর একটি করে এড্রেস রয়েছে। আর প্রতিটা বাইটে ৮টি করে বিট রয়েছে।



আমরা যখন বলি আমাদের র‍্যাম 8 Giga byte, তখন আমাদের কম্পিউটারের র‍্যামে মোট 8 000 000 000 bytes ডেটা স্টোর করা যাবে, এবং এদের প্রত্যেকের একটি করে এড্রেস রয়েছে। প্রথমটি ০ পরের টি 1, এর পরের টির এড্রেস 2 এভাবে বাড়তে থাকে। যদিও কম্পিউটার এ এড্রেস গুলো রিপ্রেজেন্ট করে হেক্সাডেসিমেল নাম্বার সিস্টেমে।

আমরা যখন একটি ভ্যারিয়েবল ডিক্লেয়ার করার পর যখন প্রোগ্রামটি এক্সিকিউট/রান করি তখন কম্পিউটার ঐ ভ্যারিয়েবল এর জন্য কিছু মেমরি এলোকেট করে। কত বাইট মেমরি এলোকেট করবে, তা নির্ভর করে ঐ ভ্যারিয়েবল এর ডেটা টাইপ এবং কম্পাইলার এর উপর।

সাধারনত কম্পাইলার গুলো একটা int এর জন্য 2 byte মেমরি এলোকেট করে। তেমনি একটি char ভ্যারিয়েবলের জন্য 1 byte মেমরি এলোকেট করে। floating-point নাম্বার এর জন্য 4 byte মেমরি এলোকেট করে।

যেমন যখন কম্পিউটার দেখে এমন একটি ডিক্লারেশন int a; তখন এটি বুঝতে পারে এটি একটি ইন্টিজার ভ্যারিয়েবল এবং এর জন্য ২ বাইট মেমরি এলোকেট করা দরকার। তখন র‍্যাম এর খালি যায়গা থেকে এটি এই ইন্টিজারের জন্য ২ বাইট মেমরি এলোকেট করে।

আমরা সহজেই একটি ভ্যারিয়েবলের মেমরি লোকেশন বের করতে পারি, নিচের প্রোগ্রামটি দেখা যাকঃ

```
#include <stdio.h>
int main()
{
int a =5;
printf("Memory address of variable a is: %x",&a);
return 0;
}
```

উপরের প্রোগ্রামটি রান করালে এমন কিছু দেখাবেঃ Memory address of variable a is: 2686732 । এক কম্পিউটারে এক এক মান দেখাবে। এবং একবার এক এক ভ্যালু দেখাবে। কারণ যতবারই আমরা প্রোগ্রামটি রান করি, প্রতিবারই ভ্যারিয়েবলটির জন্য মেমরিতে একটা জায়গা বরাদ্ধ করা হয়। আর ঐ জায়গার এড্রেসটা প্রতিবারই পরিবর্তন হয়।

কোন ভ্যারিয়েবল এর এর মেমরি এড্রেস জানার জন্য & [ampersend] ব্যবহার করা হয়। যাকে address-of operator [&] ও বলা হয়। যা দিয়ে আমরা অন্য একটি ভ্যারিয়েবল এর এড্রেস বা মেমরি লোকেশন পেতে পারি।

যখন আমরা প্রোগ্রামটি রান করি, তখন কম্পিউটার র‍্যাম এর খালি যায়গা থেকে ভ্যারিয়েবল a এর জন্য ২ বাইট মেমরি এলোকেট করে। কম্পিউটার অটোমেটিকেলি তখন a এর জন্য 2686732 এবং 2686733 নং সেল এলোকেট করে রাখে। আর মেমরি এড্রেস জানার জন্য শুধু মাত্র শুরুর এড্রেস জানলেই হয়। আমরা যখন a এর মেমরি এড্রেস প্রিন্ট করেছি, তখন শুধু

শুরুর এড্রেস 2686732 ই পেয়েছি। যদি ও a ভ্যারিয়েবল এর জন্য 2686732 এবং 2686733 মেমরি এলোকেট করা হয়েছে এবং এর মান 5 এই দুই সেলে স্টোর করে রাখা হয়েছে। এখন আমরা যদি a এর মান পরিবর্তন করে অন্য আরেকটা ভ্যালু রাখি, যেমন 8, তখন র‍্যামের 2686732 এবং 2686733 এ দুটো সেল এর মান ও পরিবর্তন হয়ে যাবে এবং এ দুটো সেলে 5 এর পরিবর্থে 8 স্টোর হবে। এবার পয়েন্টার কি জানা যাক।

পয়েন্টার

পয়েন্টার হচ্ছে একটা ভ্যারিয়েবল যার ভ্যালু হচ্ছে আরেকটি ভ্যারিয়েবল এর মেমরি এড্রেস। পয়েন্টার একটা ডেটা, অ্যারে বা ভ্যারিয়েবল এর কম্পিউটার মেমরি এড্রেস রিপ্রেজেন্ট করে বা পয়েন্ট করে। অন্যান্য ভ্যারিয়েবল এর মত পয়েন্টার ভ্যারিয়েবল ব্যবহার করার আগে কম্পিউটার/ কম্পাইলারকে বলতে হবে এটা একটি পয়েন্টার ভ্যারিয়েবল। নিচের মত করে একটি পয়েন্টার ভ্যারিয়েবল ডিক্লেয়ার করে।

data_type *name;

যেমন integer পয়েন্টারের জন্যঃ int *i;

asterisk [*] একটি ভ্যারিয়েবলের আগে ব্যবহার করে পয়েন্টার হিসেবে ডিক্লেয়ার করা হয়। যাকে indirection operator বা value-at-address operator বলা হয়। এখানে আরো কিছু ডেটা টাইপ এর পয়েন্টার ডিক্লারেশন এর উদাহরন দেওয়া হলোঃ

```
int*ip;/* pointer to an integer */
double*dp;/* pointer to a double */
float*fp;/* pointer to a float */
char*ch /* pointer to a character */
```

আমরা এখন দেখব কিভাবে পয়েন্টার ব্যবহার করতে হয় একটি প্রোগ্রামে।

```
#include <stdio.h>
int main ()
{
int a = 5; /* variable declaration */
int *ip; /* pointer variable declaration */
ip = &a; /* store address of "a" in pointer variable*/
printf("Address of a variable: %xn", &a );
/* address stored in pointer variable */
printf("Address stored in ip variable: %xn", ip );
return 0;
}
```

এখানে আমরা একটি ভ্যারিয়েবল a ডিক্লেয়ার করেছি। এরপর একটি পয়েন্টার ভ্যারিয়েবল ডিক্লেয়ার করেছি। তারপর পয়েন্টার ভ্যারিয়েবলে a এর মেমরি এড্রেস রেখেছি। তারপর &

অপারেটর দিয়ে a ভ্যারিয়েবল এর এড্রেস প্রিন্ট করে দেখলাম। এবং পয়েন্টার ভ্যারিয়েবল এর ভ্যালু প্রিন্ট করে দেখলাম। উভয় এর মান ই একই।

আমরা ইচ্ছে করলে এখন ip পয়েন্টার ভ্যারিয়েবল দিয়ে a এর মান বের করতে পারি।

```
#include <stdio.h>
int main ()
{
int a = 5;
int *ip;
ip = &a;
/* access the value using the pointer */
printf("Value of *ip variable: %dn", *ip );
return 0;
}
```

আমরা যখন প্রগ্রামটি রান করব, তখন ip যে ভ্যারিয়েবলটির এড্রেস শো করবে, তার মান প্রিন্ট করবে। লক্ষকরি, যখন আমরা পয়েন্টার ভ্যারিয়েবল দিয়ে কোন ভ্যারিয়েবল এর এড্রেস বের করতে চাইবো, তখন শুধু পয়েন্টার ভ্যারিয়েবল লিখলেই হবে। কিন্তু যখন আমরা পয়েন্টার ভ্যারিয়েবল দিয়ে মূল ভ্যারিয়েবল এর ভ্যালু বের করতে চাইবো, তখন পয়েন্টার ভ্যারিয়েবল এর আগে * যোগ করতে হবে। যেমন প্রথম প্রোগ্রামে আমরা ip [পয়েন্টার ভ্যারিয়েবল] প্রিন্ট করায় আমরা এড্রেস পেয়েছি। এবং পরের প্রোগ্রামে ip এর আগে একটা * দিয়ে *ip প্রিন্ট করায় আমরা মূল ভ্যারিয়েবলের মান পেয়েছি।

ফাংশনে পয়েন্টার পাস করা

আমরা একটি ভ্যারিয়েবলের মত একটি পয়েন্টার বা একটি ভ্যারিয়েবলের মেমরি অ্যাড্রেস ফাংশনে পাস করতে পারি। নিচের প্রোগ্রামটি দেখুনঃ

```
#include <stdio.h>

void func1(int *pNum) {
   *pNum = 5;
   return;
}

int main () {

   int num = 1;

   printf("Before passing: %d\n", num );

   func1( &num );

   printf("After passing: %d\n", num );

   return 0;
}
```

এখানে num নামে একটা ভ্যারিয়েবল নিয়েছি যার মধ্যে এসাইন করেছি 1। এরপর তা প্রিন্ট করেছি। 1 ই পেয়েছি।

এখন আমরা ফাংশনটিকে কল করেছি। কল করার সময় পাস করেছি num ভ্যারিয়েবলটির মেমরি অ্যাড্রেস। এরপর ফাংশনে গিয়ে ঐ এড্রেসের ভ্যালু পরিবর্তন করে 5 এসাইন করে দিয়েছি।

এরপর আবার num ভ্যারিয়েবলটি প্রিন্ট করেছি। আমরা পেয়েছি 5। এভাবে আমরা পয়েন্টারকেও ফাংশনে পাস করতে পারি।