পয়েন্টার হচ্ছে একটি বিশেষ ধরনের ভেরিয়েবল যেটি আরেকটি ভেরিয়েবল এর ঠিকানা রাখতে পারে।

- যে পয়েন্টার সে একটি ভেরিয়েবল এর ঠিকানা রাখবে এবং তার আগে * বসিয়ে দিলে সেই ভেরিয়েবল এর মানও পাওয়া যাবে।
- অর্থাৎ p তে যদি আমি a নামক ভেরিয়েবল রাখি তাহলে a এর মান a ব্যবহার করেও পাওয়া যাবে আবার *p ব্যবহার করেও পাওয়া যাবে।

ইন্টিজার এর জন্য ইন্টিজার পয়েন্টার। ডাবল এর জন্য ডাবল, ফ্লোট এর জন্য ফ্লোট, ক্যারেক্টার এর জন্য ক্যারেক্টার টাইপের পয়েন্টার। অর্থাৎ ভেরিয়েবল যে টাইপের হবে পয়েন্টার ও সেই টাইপের হবে।

- int *p says interger pointer p
- *p says content of p.
- called dereferencing operator

2.1

```c
#include <stdio.h>

int main()
{
	int a = 10;
	printf("Value of a : %d\n", a);
	printf("Address of a : %p\n", &a);
}
```

```
Output:
Value of a : 10
Address of a : 00D7FBA8
/*Without & address of a : 0000000A*/
```

2.2

```c
#include <stdio.h>

int main()
{
	int a = 10;
	int *p;		/*ইন্টিজার টাইপরে পয়েন্টার ডিক্লেয়ার করলাম*/

	p = &a;		/*a নামক ইন্টিজার ভেরিয়েবল এর ঠিকানা p তে রাখলাম*/

	/*অর্থাৎ p এর মান প্রিন্ট করলে a এর ঠিকান পাবো আবার a এর মান প্রিন্ট করার জন্য *p বা a প্রিন্ট করবো*/

	printf("a = %d\n", *p);
	printf("Value of p is %p\n", p);
}
```

```
Output:
a = 10
Value of p is 0073F844
```

2.4

```c
#include <stdio.h>

int main()
{
	int a = 10;
	int *p;

	p = &a; /*p পয়েন্টারে a এর ঠিকানা রেখেছি বা p এর মান হবে a এর ঠিকানা*/

	printf("Value of a : %d\n", a);//a or *p

	/*p পয়েন্টার যে ঠিকানায় আছে সেই ঠিকানায় ২০ অ্যাসাইন করে দিলাম। তাই a = 20
	অর্থাৎ আমি যদি কোনো ভেরিয়েবল হই তাহলে আমার ঠিকানা হচ্ছে আমার রোল নাম্বার*/

	*p = 20;

	printf("Value of a = %d\n", a); /*a or *p*/
	printf("Address of a = %p\n", &a);
	printf("Value of p = %p\n", p); /*a এর ঠিকানায় p এর ভেলু রেখে দিলাম*/
}
```

```
Output:
Value of a : 10
Value of a = 20
Address of a = 00EFFD4C
Value of p = 00EFFD4C*
```

2.5

```c
#include <stdio.h>

int main()
{
	int a = 10;
	int *p;
	printf("Value of a : %d\n", a);

	p = &a;
	*p = 20;
	printf("Value of a : %d\n", a);

	a = 15;
	printf("Value of a : %d\n", a);
	printf("Value store at location %p is %d\n", p, *p); //*p or a
	printf("Address of a : %p\n", &a);
	printf("Value of p : %p\n", p);
}
```

```
Output:
Value of a : 10
Value of a : 20
Value of a : 15
Value store at location
00F9FBD4 is 15
Address of a : 00F9FBD4
Value of p : 00F9FBD4*
```

2.6

```c
#include <stdio.h>

int main()
{
	int a = 10;
	int b;
	int *p;
	printf("Value of a   : %d\n", a);

	p = &a; /*p এর মান হবে a এর ঠিকানা। অর্থাৎ a এর যা মান হবে p এরও একই মান হবে*/
	b = *p;
	*p = 15;

	printf("Value of a   : %d\n", a);
	printf("Value of b   : %d\n", b);
	printf("Value of *p  : %d\n", *p);
	printf("Address of x : %d\n", &a);
	printf("Address of y : %d\n", &b);
	printf("Value of p   : %d\n", p);
}
```

```
Output:
Value of a   : 10
Value of a   : 15
Value of b   : 10
Value of * p : 15
Address of x : 15726524
Address of y : 15726512
Value of p   : 15726524*
```

2.7

```c
#include <stdio.h>

int main()
{
	int a = 10, b;
	int *p, *q;

	p = &a; /*p এর মান হবে a এর ঠিকানা। অর্থাৎ a এর যা মান হবে p এরও একই মান হবে*/
	b = *p;
	*p = 15;
	q = &b;

	printf("Value of a  : %d\n", a); //a or *p
	printf("Value of b  : %d\n", b);
	printf("Value of *p : %d\n", *p);
	printf("Value of *q : %d\n", *q);
}
```

```
Output:
Value of a   : 15
Value of b   : 10
Value of  *p : 15
Value of  *q : 10
```

2.8

```c
#include <stdio.h>

int main()
{
	int a = 10, b;
	int *p, *q;

	p = &a;
	q = &b;
	b = *p;
	*p = 15;
	*q = 20;

	printf("Value of a  : %d\n", a);
	printf("Value of b  : %d\n", b);
	printf("Value of *p : %d\n", *p);
	printf("Value of *q : %d\n", *q);
}
```

```
Output:
Value of a  : 15
Value of b  : 20
Value of *p : 15
Value of *q : 20
```

2.10

```c
#include <stdio.h>

int main()
{
	int a = 100;
	int *p = "NULL";

	printf("Value of a  : %d\n", a);
	p = &a;
	printf("Value of *p : %d\n", *p); //*p or a
}
```

```
Output:
Value of a  : 100
Value of *p : 100
```

2.12

```c
#include <stdio.h>

int main()
{
	char str[] = "Bangladesh";

	/*স্টিং হচ্ছে ক্যারেক্টারের অ্যারে আর অ্যারের নামটিই তার অ্যাড্রেস বা ঠিকানা নির্দেশ করে, তাই সেটিও প্রিন্ট করেছি।*/
	printf("Name of our country : %s\n", str);
	printf("Address of str : %p\n", str);
}
```

```
Output:
Name of our country : Bangladesh
Address of str : 00EFFA1C
```

2.13

```c
#include <stdio.h>

int main()
{
	char str[] = "Bangladesh";
	char *p;
	//or, char *p = "Bangladesh;
	p = str;

	printf("Name of our country : %s\n", p);
}
```

```
Output:
Name of our country : Bangladesh
```

2.14

```c
#include <stdio.h>

int main()
{
	char c1 = 'A', c2 = 'B', c3 = 'C';
	char *p1, *p2, *p3;

	p1 = &c1; p2 = &c2; p3 = &c3;

	printf("%c %c %c\n", *p1, *p2, *p3);
}
```

```
Output: A B C
```

2.16

```c
#include <stdio.h>

int main()
{
	char c = 'A';
	char *p, **q;

	p = &c;
	q = &p;

	printf("Value of c : %c\n", c);
	printf("Value of c : %c\n", *p);
	printf("Value of c : %c\n", **q);
}
```

```
Output:
Value of c : A
Value of c : A
Value of c : A
```

2.17

```c
#include <stdio.h>

int main()
{
	char c = 'A';
	char *p, **q;

	p = &c;
	q = &p;

	**q = 'B';

	/*c এর অ্যাড্রেস p এর মধ্যে আবার p এর অ্যাড্রেস q এর মধ্যে*/
	printf("Value of c : %c\n", c);
	printf("Value of c : %c\n", *p);
	printf("Value of c : %c\n", **q);
}
```

```
Output:
Value of c : B
Value of c : B
Value of c : B
```

ডায়নামিক মেমোরি অ্যালোকেশন যা ব্যবহার করলে আমাদের আগে থেকে মেমোরির সাইজ নির্ধারন করতে হবে না। প্রোগ্রামটি চলার সময়ই আমরা দরকার মতো মেমোরি নিতে পারবো। এজন্য পয়েন্টার ব্যবহার করে আমাদের মেমোরি বরাদ্দ বা অ্যালোকেশন করেত হবে।

7.1

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
	/*এখানে অ্যারে ব্যবহার না করে ইন্টিজার পয়েন্টার ব্যবহার করেছি। কারন ইন্টিজার পয়েন্টার কেবল ইন্টিজারকের পয়েন্ট করতে পারে এমন নয় এটি ইন্টিজার অ্যারেকেও পয়েন্ট করতে পারে।*/
	int *marks;
	int i, n;
	printf("Enter the number of sutudents : ");
	scanf("%d", &n);

	/*প্রতিটি নম্বর হচ্ছে ইন্টিজার আর মোট ইন্টজার সংখ্যা n তাই আমাদের sizeof(int) x n বাইট জায়গা প্রয়োজন।*/

	marks = (int *)malloc(sizeof(int) * n);

	/*ঠিকঠাক মেমোরি বরাদ্দ করতে পারলে malloc ফাংশনটি একটি পয়েন্টার রির্টান করবে, যে মেমোরি বরাদ্দ করা হয়েছে তার শুরুর ঠিকানা। আর কোনো সমস্যা হলে NULL রির্টান করবে। (int *) লিখলাম কারন *marks হচ্ছে ইন্টজার পয়েন্টার।*/

	printf("Enter the marks for each students : \n");
	for (i = 0; i < n; i++) {
		scanf("%d", &marks[i]);
	}

	printf("The marks is : \n");
	for (i = 0; i < n; i++) {
		printf("%d ", marks[i]);
	}
}
```

```
Output:
Enter the number of sutudents : 5
Enter the marks for each students :
67 78 89 87 70
The marks is :
67 78 89 87 70
```

malloc এর মতো আরেকটি লাইব্রেরি ফাংশন হচ্ছে calloc. এটি malloc এর মতোই কাজ করে তবে একটু পাকনামি বেশি করে এটাই সমস্যা। এটি মেমোরির যতটুকু অংশ বরাদ্দ করে হয়েছে সেখানে সব 0 দিয়ে ইনিশিয়ালাইজ করে দেয়।

/*আগের প্রোগ্রামটিকেই আরেকটু সাজিয়ে লিখলাম*/
7.2

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
	int *marks;
	int i, n;
	printf("Enter the number of students : ");
	scanf("%d", &n);

	marks = (int *)malloc(sizeof(int) * n);
	if (marks == NULL) {
		printf("Memory allocation failed for marks\n");
		return 1;
	}

	printf("Enter the marks of the students : \n");
	for (i = 0; i < n; i++) {
		scanf("%d", &marks[i]);
	}

	printf("The marks is : \n");
	for (i = 0; i < n; i++) {
		printf("%d ", marks[i]);
	}

	/*Now free the memory*/
	free(marks);
}
```

```
Output:
Enter the number of students : 5
Enter the marks of the students :
55 66 77 76 89
The marks is :
55 66 77 76 89
```

7.3

/*প্রথম হতে তৃতীয় শ্রেনী পর্যন্ত সব শিক্ষার্থীর গণিত পরীক্ষার নম্বর একটি অ্যারেতে রাখতে চাচ্ছি তারমানে marks[3][1000] অর্থাৎ প্রতি শ্রেনিতে সর্বোচ্চ 1000 জন করে। কিন্ত ডায়নামিক মেমোরি অ্যালোকেশন এর মাধ্যেমে একটি পয়েন্টার অ্যারে তৈরি করে নিম্নোক্ত ভাবে করতে পারি*/

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
	int *array[3], num[3];
	int i, j, n;

	for (i = 0; i < 3; i++)
	{
		printf("Enter number of students for class %d : ", i + 1);
		scanf("%d", &n);
		num[i] = n;
		array[i] = (int*)malloc(sizeof(int) * n);
		if (array[i] == NULL) {
			printf("Memory allocation failed\n");
			return 1;
		}
		for (j = 0; j < n; j++) {
			printf("Enter marks for student %d : ", j + 1);
			scanf("%d", &array[i][j]);
		}
	}

	printf("Result\n");
	for (i = 0; i < 3; i++)
	{
		printf("Class %d : \n", i + 1);
		for (j = 0; j < num[i]; j++) {
			printf("%d\n", array[i][j]);
		}
		printf("\n");
	}
	//free(array);
}
```

```
Output:
Enter number of students for class 1 : 5
Enter marks for student 1 : 67
Enter marks for student 2 : 78
Enter marks for student 3 : 89
Enter marks for student 4 : 98
Enter marks for student 5 : 84
Enter number of students for class 2 : 3
Enter marks for student 1 : 78
Enter marks for student 2 : 99
Enter marks for student 3 : 90
Enter number of students for class 3 : 2
Enter marks for student 1 : 98
Enter marks for student 2 : 96
```

```
Result
Class 1 :
67
78
89
98
84

Class 2 :
78
99
90

Class 3 :
98
96
```

কিছু বেসিক বিষয়ঃ

int *p; একটি ইন্টিজার টাইপের পয়েন্টার ডিক্লেয়ার করলাম।

- p - একটি ইন্টিজার এর মেমোরির অ্যাড্রেস রাখতে পারবো।
- *p - সেই মেমোরি অ্যাড্রেস এ যে ভেলুটি আছে সেটি প্রিন্ট করতে পারবো।
- *p+1 - বলতে বুঝায় p এর ঠিকানায় যে ভেরিয়েবল আছে তার মান এর সাথে এক যোগ।
- *(p+1) - p এর ঠিকানার পরবর্তী ঠিকানায় যে ভেরিয়েবল আছে তার মান।

7.5

```c
#include <stdio.h>

int main()
{
	int num[] = { 100, 300, 500, 700, 900 };
	int *p;
	p = num;

	printf("*p     = %d\n", *p);
	printf("*p+1   = %d\n", *p + 1);
	printf("*(P+1) = %d\n", *(p + 1));
	printf("*p+2   = %d\n", *p + 2);
	printf("*(p+2) = %d\n", *(p + 2));
}
```

```
Output:
*p       = 100
*p + 1   = 101
*(P + 1) = 300
*p + 2   = 102
*(p + 2) = 500
```

7.6

```c
#include <stdio.h>

int main()
{
	char *str = "Bangladesh";

	printf("%c %c %c %c\n", *str, *(str + 1), *(str + 2), *(str + 3));
	printf("%c %c %c %c\n", *str, *str + 1, *str + 2, *str + 3);
}
```

```
Output:
B a n g
B C D E
```

7.7

```c
#include <stdio.h>

int main()
{
	char *p, a = 10;
	int *q, b = 'F';
	double *r, c = 302.64;

	p = &a; q = &b; r = &c;

	printf("Size of character : %d bytes\n", sizeof(char));
	printf("p   = %p\n", p);
	printf("p+1 = %p\n", p + 1);
	printf("p+2 = %p\n", p + 2);

	printf("Size of interger  : %d bytes\n", sizeof(int));
	printf("q   = %p\n", q);
	printf("q+1 = %p\n", q + 1);
	printf("q+2 = %p\n", q + 2);

	printf("Size of double    : %d bytes\n", sizeof(double));
	printf("r   = %p\n", r);
	printf("r+1 = %p\n", r + 1);
	printf("r+2 = %p\n", r + 2);
}
```

```
Output:
Size of character : 1 bytes
p     = 00EFFC9F
p + 1 = 00EFFCA0
p + 2 = 00EFFCA1
Size of interger : 4 bytes
q     = 00EFFC84
q + 1 = 00EFFC88
q + 2 = 00EFFC8C
Size of double : 8 bytes
r     = 00EFFC68
r + 1 = 00EFFC70
r + 2 = 00EFFC78
```

Void pointer:

void pointer ডিক্লেয়ার করার নিয়মঃ void *ptr;

- যদি n একটি ইন্টিজার ভেরিয়েবল হয় তাহলে তার অ্যাড্রেস আমরা একটি পয়েন্টার এর মধ্যে রাখতে পারবো এইভাবেঃ ptr = &n;
- আবার, আমরা যদি ptr কে ডিরেফারেন্স করতে চাই তাহলে শুধু *ptr লিখলেই হবে না আমাদের টাইপ কাষ্ট করে নিতে হবে এভাবেঃ *((int*)ptr)

7.8

```c
#include <stdio.h>

int main()
{
	void *vp; //vp means void pointer.
	int n = 10;

	vp = &n;

	printf("Address of n  = %p\n", &n);
	printf("Value of vp   = %p\n", vp);
	printf("Content of vp = %d\n", *((int*)vp));
}
```

```
Output:
Address of n  = 012FFB88
Value of vp   = 012FFB88
Content of vp = 10
```

```c
#include <stdio.h>

int main()
{
	int a = 5;
	int *p;   /*Pointer declaration*/
	p = &a;   /*Copying address of a to pointer p*/
	*p = 10;  /*Use pointer to change the value of variable a*/

	printf("%d, ", a);
	printf("%d, ", *p);
	printf("%d", *&a);
}
```

```
Output: 10, 10, 10
```

```c
#include <stdio.h>

int main()
{
	int a = 5, *p1;
	float b = 2.5, *p2;
	char c = 'a', *p3;

	p1 = &a;
	p2 = &b;
	p3 = &c;

	printf("%d, ", sizeof(p1));
	printf("%d, ", sizeof(p2));
	printf("%d", sizeof(p3));
}
```

```
Output: 4, 4, 4
```

```c
/*Any type of pointer gets two bytes in the memory*/
/*This output is depend on the computer. My compiler gives 4, 4, 4*/
/*Pointer store the address of a variable*/
```

```c
#include <stdio.h>

int main()
{
	int a[5] = { 10, 20, 30, 40, 50 }, * p;
	p = &a;
	/*Address of array can be pointed into the pointer without the use of &
operator like that p = a*/
	printf("%d, ", *++p);	/*Point index number 1*/
	printf("%d, ", ++*p);
	printf("%d, ", *p--);
	printf("%d", *p);
}
```

```
Output:
20, 21, 21, 10
```

```c
#include <stdio.h>

int main()
{
	int a = 5;
	int *p1;	/*Pointer to an integer*/
	int **p2;	/*Pointer to pointer to an integer*/
	int ***p3;	/*Pointer to pointer to pointer to an integer*/

	p1 = &a;
	p2 = &p1;
	p3 = &p2;

	printf("%d, ", a);
	printf("%d, ", *p1);
	printf("%d, ", **p2);
	printf("%d, ", ***p3);
}
```

```
Output:
5, 5, 5, 5
```

- Two pointer can not be multiplied divide or added.
- NULL pointer means not to point anywhere.

A NULL pointer is declare many ways which is given below:

- int *p = NULL;
- int *p = 0;
- if (P != NULL) printf("%d", *p);
- if (p) printf("%d", *p);

/*This has the same meaning as our previous example;
if (p) is equivalent to if (p!= 0) and to if (p != NULL)*/

```c
/*Swap two numbers*/

#include <stdio.h>

int main()
{
	int a, b;
	printf("Before swapping a and b : ");
	scanf("%d %d", &a, &b);

	int temp;
	temp = a;
	a = b;
	b = temp;

	printf("After swapping a and b  : %d %d\n", a, b);
}
```

```
Output:
Before swapping a and b : 5 10
After swapping a and b  : 10 5
```

```c
/*Swap two numbers using pointer*/

#include <stdio.h>

int main()
{
	void swap(int *x, int *y);

	int a, b;
	printf("Before swapping a and b : ");
	scanf("%d %d", &a, &b);

	swap(&a, &b);
	printf("After swapping a and b  : %d %d", a, b);
}
void swap(int *x, int *y)
{
	int temp;
	temp = *x;
	*x = *y;
	*y = temp;
}
```

```
Output:
Before swapping aand b : 5 10
After swapping a and b : 10 5
/*The function prototype is declared as pointers.*/
/*When the function is called, the addresses are passed as actual arguments.*/
```

```c
/*Write a program to extract a substring from a string*/

#include<stdio.h>
#include<string.h>

int main()
{
	char str[20], news[20];
	char *s, *t;
	int position, n, i;
	printf("Enter the string:");
	gets(str);
	printf("Enter the position and number of characters to extract:");
	scanf("%d %d", &position, &n);
	s = str;
	t = news;
	if (n == 0) n = strlen(str);
	s = s + position - 1;
	for (i = 0; i < n; i++)
	{
		*t = *s;
		s++;
		t++;
	}
	*t = '\0';
	printf("The substring is: %s\n", news);
}
```

```
Output:
Enter the string : Dhaka bangladesh
Enter the positionand number of characters to extract : 3 8
The substring is : aka bang
```