C Programming in Sinhala

by Deshan Nawanjana
http://bracketslk.blogspot.com/

1. First Program

1_first_program.c

```
void main() {

}
```

C භාෂාවෙන් ලිවි හැකි කෙටිම සහ සරළම මෙන්ම, C පරිගණක වැඩසටහනක තිබිය යුතු අවශ්‍යම කොටස වන්නේද මෙයයි. මෙය Main Function එක වන අතර වැඩසටහන ක්‍රියාත්මක වීමේදී මේ තුළ ඇති කේත මූලික වශයෙන් එකින් එක පහළට ධාවනය වීම සිදු වේ.

2. Variables

2_variables.c

```
void main() {
        // variable deceleration width 'garbage values'
        int x;   // integer
        char c;  // character
        float y; // floating point

        // assigning values to variables
        x = 12;
        c = 'K';
        y = 4.5; // becomes 4.500000

        // variable deceleration and value assigning same time
        int a = 20;
        char d = 'f';

        // constant variable deceleration
        const int z = 10;
        // z = 20 is wrong! z is read only
}
```

Variables යනු දත්ත මතකයේ තබා ගන්නා ඒකක වේ. C භාෂාවේ මූලික විචල්‍ය වර්ග තුනක් වන අතර ඒවා Integer (නිඛිල සංඛ්‍යා), Float (දශමය සංඛ්‍යා) සහ Charactors (අක්ෂර හෝ සංඛේත) වේ. මෙහිදී අප වටහාගත යුතු දෙයක් වන්නේ විචළල්‍යයක් තැනීමේදී සිදු වන්නේ පරිගණකයේ RAM එක මත ඇති දහස්ගානක Memory Blocksවලින් එක Block එකක් මේ විචල්‍යයක් සඳහා වෙන් වන බවයි. විචල්‍යයක් තැනීමේදී එයට අගයක් ඉදිරියෙන් ඒ මොහොතේම ලබා නෙදේ නම්, එම විචල්‍යයට ඇත්තේ Garbage Value එකකි. එනම් අදාළ Memory Block එක පෙර වැඩසටහනකින් භාවිත කර ඇත්නම් එම වැඩසටහනේ කාර්යය අවසන් වූ පසු එහි ඉතුරු කර ගොස් ඇති අගයයි. එනම් Interger විචල්‍යයක් තැනූ සැනින් එහි අගය 0 වන්නේ නැත. අපට කැමති අවස්ථාව මෙම විචල්‍යයන්ගේ අගයන් වෙනස් කළ හැකියි. Float සඳහා දශම සංඛ්‍යා තැන්පත් කරන අතර එහි සෑම විටම දශම ස්ථාන 6ක් සහිතව අදාළ අගය ගබඩා වේ. Char සඳහා අක්ෂරයක් හෝ සංඛේතයක් ලබා දීමේදී තනි උඩු කොමා භාවි වේ. තවද const නම් Keyword මගින් තැනූ විචල්‍යයක අගය එය නිර්මානය කරන අවස්ථාවේ ලබාදිය යුතු අතර, එම අගය පසුවට වෙනස් කිරීමට නොහැක.

3. Include

3_include.c

```
#include <stdio.h>  // for standard input and output
#include <math.h>   // for mathematical operations
#include <time.h>   // for deal with time calculations
#include <string.h>   // for deal with strings [charactor arrays]

void main() {
        /*
                printf();
                scanf();

                fopen();
                fprintf();
                fscanf();
                fclose();

                strlen();
                strcmp();
                strcat();

        */
}
```

#include මගින් අප වැඩසටහනට අවශ්‍ය බෙහෝ Functions එක් කර ගැනීම සිදු වේ. Functions පිළිබඳ පසුව විස්තරාත්මකව ඉදිරිපත් කර ඇත. එම කොටස්වලදී මෙය වැඩි වශයෙන් තේරුම් ගැනීමට හැකි වනු ඇත.

4.1. Printf Begin

4_printf_1_begin.c

```
#include <stdio.h>

void main() {
        printf("Hello World!");
}
```

මෙම C වැඩසටහන් ක්‍රියාත්මක වන්නේ Windows Command Prompt එක මතය. එබැවින් Command Prompt එක මත අපට අවශ්‍ය ආදාන හා ප්‍රතිදාන සිදු කරනු ලැබේ. අපට අවශ්‍ය කිසියම් ආකාරයෙක වැකියක් දර්ශනය කිරීමට printf() නම් Function එක භාවිත කෙරේ. මෙහිදී printf() යන්න stdio.h නම් Librabry File එක තුල හඳුන්වා දී ඇති බැවින් එම ගොනුව මෙම වැඩසටහනට ඇතුලත් කළ යුතුය. printf() තුළ අපට අවශ්‍ය වැකිය ද්විත්ව උඩුකොමා සහිතව දක්වනු ලැබේ.

4.2. Printf Escape Charactor

4_printf_2_escape_charactor.c

```
#include <stdio.h>

/*

        \n = New Line
        \t = Tab Space
        \' = Single Quote
        \" = Double Quote
        \\ = Backslash

*/

void main() {
        printf("Text 1\nText 2\n");
        printf("Text 1\tText 2");
        printf("\n");
        printf("\'Single Quotes\'");
        printf("\n");
        printf("\"Double Quotes\"");
        printf("\n");
        printf("\\ is a Backslash");
        printf("\n");
        printf("\\\\ are two Backslashes");
}
```

සාමාන්‍යයන් printf() එකක් තුළ සෘජුවම දැක්විය නොහැකි ස්ඛේත පවතී. නිදසුනක් ලෙස ද්විත්ව උඩුකොමාව ගත හැකිය. මක් නිසාදයත් අපට දර්ශනය කිරීමට අවශ්‍ය වැකිය දෙපස ද්විත්ව උඩු කොමා ඇති බැවින් එහි මැදට නැවත ද්විත්ව උඩුකොමාවක් එකවර යෙදිය නොහැක. මේ නිසා භාවිතා වන Escape Charactor එකක් ලෙස පසු ඇල ඉර (\) හැඳින්විය හැකිය. ඇල ඉරක් සමග පසුවට යෙදෙන අක්ෂරය හෝ සංඛේතය මත එම ස්ථානයේ නිරූපණය විය යුතු ස්ඛේතය දර්ශනය වේ. Main Function එකට ඉහළින් ඇති Comment එක තුළ අදාළ ආකාර දක්වා ඇත. නිදසුනක් ලෙස printf() දෙකක් පහළට ලබා දුන්නේ යැයි කියා එම වැකි දෙක වෙන වෙනම පේළි දෙකක දර්ශනය නොවේ. එසේ වීමට නම් New Line Charactor එකක් යෙදීමට අවශ්‍යය. එය \n මගින් අපට අවශ්‍ය තැනට ලබා දිය යුතුය. එමෙන්ම පසු ඇල ඉරි දෙකක් එක ලඟ යෙදීමෙන් එක පසු ඇල ඉරක් දර්ශනය කළ හැක.

4.3. Printf Variables

4_printf_3_variables.c

```c
#include <stdio.h>

/*
        %d = integers
        %f = floats
        %c = charactors
        %s = strings
        %p = pointers
*/

void main() {
        int x = 10;
        printf("value of x is %d\n", x);

        float y = 25.2;
        printf("value of y is %f\n", y);

        char z = 'w';
        printf("value of z is %c\n", z);

        printf("x = %d, y = %f, z = %c\n", x, y, z);
}
```

printf() මගින් අපට අවශ්‍ය වැකි පමණක් නොව බොහෝ විට විචල්‍ය අගයන් පවා දැක්වීමට සිදු වේ. ඒ සඳහා අප % නම් Escape Charactor එක භාවිතා කරයි. මෙහිදී printf() වෙත දෙන වැකියේ විචල්‍ය අගය දැක්විය යුතු තැනට % ලකුණ සහිතව විචල්‍ය වර්ගයට අනුරූප අක්ෂරය දෙනු ලැබේ. එම අක්ෂර ලැයිස්තුව නිදසුනේ Comment කර ඇත. පසුව printf() හි අප දෙන වැකිය අවසානයේ කොමා දක්වමින් අදාළ විචල්‍ය පිළිවෙලට ලබා දීම සිදු කළ යුතුය. මෙදිහිදී ලබා දෙන පිලිවෙල වැදගත් වේ. එක printf() එකක් තුළ එක විචල්‍ය නිරූපණයක් පමණක් නොව අපට අවශ්‍ය තරම් නිරූපණය කිරීම කළ හැක.

4.4. Printf Values

4_printf_4_values.c

```c
#include <stdio.h>

/*
        %d = integers
        %f = floats
        %c = characters
        %s = strings
        %p = pointers
*/

void main() {
        printf("My name is %s.\n", "Kamal");
        printf("I am %d years old.\n", 22);
        printf("My average is %f%\n", 83.4);
        printf("My blood group is %c Positive\n", 'O');
}
```

printf() වෙත අපට විචල්‍ය ලබා දී ඒවායේ අගයන් නිරූපණය කළ හැකි මෙන්ම විචල්‍ය පමණක් නොව අගයක් හෝ ලබා දිය හැකිය. මෙ බැලූ බැල්මට වැදගත්කමක් නැති දෙයක් යැයි සිතුනත්, පසුවට මෙහි භාවිත අවස්ථා ඔබට වැටහෙනු ඇත. මෙහිදීද ඔබ ලබා දෙන අගයේ දත්ත වර්ගයට අනුරූපව Escape Charactor එක සමඟ එන අක්ෂරය ලබා දිය යුතුය.

4.5.1. Printf Align And Spacing

4_printf_5_align_and_spacing_1.c

```
#include <stdio.h>

/*
	%{align_size}.{float_sort / part_length}d
*/

void main() {
	float a = 65.797;  //  65.700000
	float b = 723.633; // 723.630000
	float c = 12.5224; //  12.522000
	float d = 8.4567;  //   8.450000

	// common represent
	printf("%f\n%f\n%f\n%f\n", a, b, c, d);

	printf("\n");

	// aligned to right with 12 blocks
	printf("%12f\n%12f\n%12f\n%12f\n", a, b, c, d);

	printf("\n");

	// aligned to left with 12 blocks
	printf("%-12f\n%-12f\n%-12f\n%-12f\n", a, b, c, d);

	printf("\n");

	// sorting to 2 floating points
	printf("%.2f\n%.2f\n%.2f\n%.2f\n", a, b, c, d);

	printf("\n");

	// aligned to right with 12 blocks and sorting to 2 floating points
	printf("%12.2f\n%12.2f\n%12.2f\n%12.2f\n", a, b, c, d);
}
```

printf() හි දක්වන වැකිය අපට අවශ්‍ය පරිදි වෙකෙල්ල කිරීම හෝ ඉඩ සහිතව දැක්වීම කළ හැක. මෙම නිදසුනේ ආකාරයට දශම සංඛ්‍යා 4ක් සැළකූ විට ඒවායේ සෑම විටම දශම ස්ථාන 6ක් පවතින නිසාත්, පූර්ණ අගය නිරූපණයට යන ඉඩ වෙනස් නිසාත් සාමාන්‍ය ලෙස පහළින් පහළට දැක්වීමේදී දශම අගයන් ස්ථාන එක පෙළට නොදැක්වේ. නමුත් අපට අවශය ලෙස ඒවා වමට හෝ දකුණට වෙකෙල්ල කිරීම කළ හැකිය. ඒ සඳහා අදාළ Escape Charactor එක හා එය සමඟ එන අක්ෂරය අතරට + හෝ - නිඛිල අගයක් ලබා දේ. එමගින් + නම් එම නිඛිල අගයට අදාළ අක්ෂර ස්ථාන ගනනක් තුළ විචල්‍ය අගය දකුණට වෙකෙල්ල වීම හෝ - නම් එම අගය වමට වෙකෙල්ල වීම සිදු වේ. වගුවක් නිර්මාණය කිරීමේදී මෙ ක්‍රමය බොහෝ සෙයින් උපකාරී වේ. එමෙන්ම දශමය අගයක් නම් අප වෙකෙල්ල කිරීමට ලබා දෙන නිඛිල අගය සමඟ දශම අගයක් ලබා දුන්නහොත් එම දශමය අගය නිරූපණය වන්නේ එම ලබා දුන් සංඛ්‍යාවට සරිලන දශම ස්ථාන ගණනකට වැටයීමෙන් පසුවය.

4.5.2. Printf Align And Spacing

4_printf_5_align_and_spacing_2.c

```
#include <stdio.h>

void main() {
        printf("%-15s %8s %8s\n", "Name", "Age", "AVG");

        printf("%-15s %8d %8.2f\n", "Kamal", 22, 65.2);
        printf("%-15s %8d %8.2f\n", "Sunimal", 23, 82.13);
        printf("%-15s %8d %8.2f\n", "Kodithuwakku", 20, 4.53);
}
```

මෙහි දැක්වෙන්නේ කලින් කතා කළ එකෙල්ල කිරීම වගුවක් නිරූපණයට භාවිත කළ ඇති ආකාරයයි. මෙහිදී පැවසිය යුතු තව දෙයක් වන්නේ Escape Charactor මගින් නිරූපණය කළ හැකිවන්නේ විචල්‍යයක් පමණක් නොව සෘජුවම අගයක් හෝ වැකියක් වුවත් ලබා දිය හැකිද බවයි.

5.1. Scanf Begin

5_scanf_1_begin.c

```
#include <stdio.h>

void main() {
        int x;

        printf("garbage value of x      = %d\n", x); // display value of x [garbage value]

        printf("enter a new value  to x : ");
        scanf("%d", &x); // &x = memory block address of variable x

        printf("new value of x          = %d\n", x); // display the new value of x
}
```

මෙහිදී අප සළකනුයේ පරිහීලකයෙන් වැඩසටහනට කිසියම් අගයක් ආදානය කරන ආකාරයයි. මේ සඳහා scanf() Function එක භාවිත වේ. මෙසේ ආදානය කරගන්නා අයගන් විචල්‍යවල ගබඩා කිරීම සිදු වේ. මෙහි එන නිදසුනේ x නමැති නිබිල විචල්‍යයක් මුලින්ම තනා ඇත. පසුව එහි Garbage Value එක කුමක්දැයි දර්ශනය කර ඇත. නැවතද printf() එකක් යොදා ඇත්තේ x සඳහා අගයක් ලබා දෙන ලෙස පරිහීලකයට දැක්වීමටයි. මෙය දත්ත ආදානයට අත්‍යාවශ්‍ය නොවුනත් වැඩසටහන තුළ දත්තයක් ආදානය කළ යුතු අවස්ථාවක එය පරිහීලකට හඳුනාගැනීමට දැක්වීම සුදුසු වේ. ඉන් පසුව ඇති scanf() මගින් දත්ත ආදානය කරගනු ඇත. මෙහිදී ද්විත්ව උඩුකොමා තුළ ආදානය කරගන්නා දත්තයේ වර්ගයට අදාළ Escape Charactor සහ අක්ෂරය යෙදිය යුතුය. පසුව කොමෝ සහිතව එම ආදානය කරගන්නා අගය තැම්පත් කළ යුතු විචල්‍යයේ RAM Memory Location Address එක ලබා දිය යුතුය. එය නිරූපණය වනුයේ විචල්‍ය නම සමග ඊට මුලට & ලකුණ යෙදීමෙනුයි. පසුව නැවත printf() මගින් නව විචල්‍ය අගය පෙන්වා දී තිබේ.

5.2. Scanf Multiple Inputs

5_scanf_2_multiple_inputs.c

```
#include <stdio.h>

void main() {
        int x, y, z;

        printf("Enter value for x : ");
        scanf("%d", &x);
        printf("Enter value for y : ");
        scanf("%d", &y);
        printf("Enter value for z : ");
        scanf("%d", &z);

        float a, b, c;

        printf("Enter float values for a, b, c : ");
        scanf("%f %f %f", &a, &b, &c);

        int k;
        char l;
        float m;

        printf("Enter values for k, l, m : ");
        scanf("%d %c %f", &k, &l, &m);

}
```

scanf() මගින් එක් වරකට එක් ආදානයක් පමණක් නොව ආදාන කිහිපයක් වුවත් ලබා ගත හැක. මෙහිදීද printf() විචල්‍ය ලබා දුන් පිලිවෙල ගැන සැලකිලිමත් වූ මෙන් විචල්‍ය ලිපින ලබා දීමේ පිලිවෙල පිලිබඳවද සැලකිලිමත් විය යුතුය. මෙම නිදසුනේ මුලින්ම වෙන වෙනම විචල්‍ය අගයන් ආදානය කරගන්නා ආකාරය දක්වා ඇති අතර පසුව දශමය ආගයන් තුනකුත්, ඉන් පසුව දත්ත ආකාර තුනෙන්ම එකවර ආදාන ලබාගන්නා ආකාරය දක්වා ඇත.

6.1. Operators

6_operators_1.c

```
#include <stdio.h>

void main() {
        int x = 25; // value assigning
        int y = 10;
        int z;

        z = x + y; // addition
        printf("z = %d\n", z);

        z = x - y; // substraction
        printf("z = %d\n", z);

        z = x * y; // multipication
        printf("z = %d\n", z);

        z = x / y; // division [no floats]
        printf("z = %d\n", z);

        float w;
        w = x / y; // division [no floats -> int/int]
        printf("z = %f\n", w);

        w = x / (float)y; // division [float -> int/float, float/int, float/float]
        printf("z = %f\n", w);

        z = x % y; // remain
        printf("z = %d\n", z);
}
```

C භාෂාවේදීද අනෙක් පරිගණක භාෂා වෙල මෙන් ගණිතකර්ම සිදු කිරීමේ සංඛේත භාවිත වේ. එකතු කිරීම සඳහා + ලකුණත්, අඩු කිරීමට - ලකුණත්, ගුණ කිරීමට * ලකුණත්, බෙදීම සඳහා / ලකුණත් භාවිත වේ. නිඛිල දෙකක් හෝ, නිඛිල විචල්‍ය දෙකක් බෙදීමේදී එහි බෙදුන අගයේ දශම අගයන් ඉතිරිවන්නේ නැත. දශම සහිත බෙදීමකට නම් බෙදීම සිදු කරන අගයන් එකක් හෝ අවම වශයෙන් දශමය විය යුතුය. එමෙන්ම එම බෙදූ අගය දශම සහිතව ගබඩා කිරීමට නම් දශම විචල්‍යයක් භාවිතා කිරීමද අත්‍යාවශ්‍යය. තවද % කර්මය මගින් යම් නිඛිල අගයන් දෙකක් බෙදුන පසු ලැබෙන ඉතිරිය ලබා දේ.

6.2. Operators

6_operators_2.c

```c
#include <stdio.h>

void main() {
        int x = 10;

        printf("x = %d\n", x);

        x = x + 1;
        printf("x = %d\n", x);

        x += 1; // x = x + 1
        printf("x = %d\n", x);

        x++; // x += 1
        printf("x = %d\n", x);

        x--; // x -= 1
        printf("x = %d\n", x);

        x -= 2; // x = x - 2
        printf("x = %d\n", x);

        x *= 5; // x = x * 5
        printf("x = %d\n", x);

        x /= 10; // x = x / 10
        printf("x = %d\n", x);
}
```

මෙම කොටසේ දැක්වෙන්නේ අමතර ගණිතකර්ම කිහිපයකි. මේවා බොහෝවිට ගණත කර්මවලට වඩා භාවිත වන්නේ විශේෂ කේත සඳහාය (For, While Loops වැනි). x = x + 1 මගින් x ට අගයක් එකතු වන අතර එය x += 1 ලෙසද ලිවිය හැක. මෙමගින් කේතය කුඩා වන අතර පහසුවෙන් හඳුනාගත හැකි වනු ඇත. අනෙකුත් මූලික ගණත කර්මද මේ ආකාරයෙන් සාරාංශ කළ හැක. x++ මගින් සිදු වනුයේ සෑම විටම xහි අගය එකකින් වැඩි වීමයි. එමෙන්ම x-- මගින් xහි අගය එකකින් අඩු වේ.

6.3. Operators

6_operators_3.c

```
#include <stdio.h>

/*

        ++x;    operator runs before the line execution
        x++;    operator runs after the line execution

*/

void main() {
        int x = 10;
        int y = 3;

        printf("X = %d, y = %d\n", x, y);

        // before x = 10, y = 3
        printf("line 1 = %d\n", (x++ * y)); // while x = 10, y = 3
        // after x = 11, y = 3

        printf("X = %d, y = %d\n", x, y);

        // before x = 11, y = 3
        printf("line 1 = %d\n", (++x * y)); // while x = 12, y = 3
        // after x = 12, y = 3

        printf("X = %d, y = %d\n", x, y);
}
```

පෙර සඳහන් කළ x++ යන්න ++x ලෙසද ලිවිය හැක. එමගින් සිදු වන්නේද x හි අගය එකකින් වැඩි වීමම වුවත් එම අවස්තා දෙකෙහි වෙනසක් ඇත. x++ යන්න කිසියම් කේත පේලියක තිබේ නම්, x හි අගය 1කින් වැඩි වනුයේ අදාළ පේලිය ධාවනය වීමෙන් අනතුරුවය. එනම් එම කේත පේලියේ ගණනය කිරීමකට ගනුයේ x හි අගය එකකින් වැඩි වීමට පෙර අගයයි. නමුත් ++x යන්න කේත පේලියක තිබුනහොත් එම පේලිය ධාවනය වන්නට පෙර x හි අගය 1කින් වැඩි වේ. x-- සහ --x යන්නද ඒ ආකාරයේද වේ.

6.4. Operators

6_operators_4.c

```
#include <stdio.h>

/*

        >               grater than
        <               less than
        ==              equal
        >=              grater than or equal
        <=              less than or equal

        1 = true
        0 = false

*/

void main() {
        int x = 10;
        int y = 5;
        int z = 10;

        printf("x > y is %d\n", x > y);
        printf("x < y is %d\n", x < y);
        printf("x == y is %d\n", x == y);
        printf("x == z is %d\n", x == z);
        printf("x >= y is %d\n", x >= y);
        printf("x <= y is %d\n", x <= y);
}
```

තවද අගයන් දෙකක් සැසඳීමටද ගණිත කර්ම කිහිපයක් පිවතී. මෙම කර්මවල ප්‍රතිදානය 1 හෝ 0 ලෙස ලැබේ. එනම් 1 යනු අදාළ සැසඳීම සත්‍ය සහ 0 යනු අසත්‍ය ලෙස ලැබේ. < සහ > මගින් අසමානතාවය සහ == මගින් සමානද යන්නත් >= සහ <= මගින් විශාල හෝ සමාන සහ කුඩා හෝ සමාන බව දැක්වේ. තවද != මගින් අසමානද යන්නත් දැක්වේ.

7. If Else

7_if_else.c

```
#include <stdio.h>

void main() {
        int x = 5;
        int y = 10;

        if(x==5) {
                printf("x is equal to 5\n");
        }

        if(x>y) {
                printf("x is larger than y\n");
        }
        else {
                printf("x is not larger than y\n");
        }

        if(x>y) {
                printf("x is larger than y\n");
        }
        else if(x<y) {
                printf("x is less than y\n");
        }
        else {
                printf("x is equal to y\n");
        }
}
```

කලින් සඳහන් කළ සැසඳීමකින් එම සැසඳීම සත්‍ය නම් යමක් කිරීමට හෝ එසේ නොමැතිව අසත්‍ය නම් වෙනත් යමක් සිදු වීමට සැලැස්විය හැකිය. ඒ සඳහා `if` කේතය භාවිතා වේ. `if()` තුළ ඇති ප්‍රකාශය සත්‍ය නම් ඒ සමගම ඇති `{}` සඟල වරහන තුළ ඇති කේත ක්‍රියාත්මක වේ. එම ප්‍රකාශය අවශ්‍ය නම් `else` කේත සමග ඇති සඟල වරහනේ කේත ක්‍රියාත්මක වේ. මෙම නිදසුනේ පළමු `if` කේතය එවැනි එකකි. එහි කේත බෙදීම් ඇත්තේ `if` සහ `else` ලෙස සමණි. නමුත් `else if` පදය මගින් අතිරේක ප්‍රකාශද ලබා දිය හැකිය එවිට එකින් එක ප්‍රකාශ පහළට සළකැ පැමිණෙන අතර සත්‍යවන ප්‍රකාශයට අදාළ කේත කොටස් ධාවනය වී ඉවත් වේ.

8.1. Cases

8_cases_1.c

```
#include <stdio.h>

void main() {
        char x = 'C';

        switch(x) {
                case 'A':
                        printf("Your Grade is A\n");
                        break;
                case 'B':
                        printf("Your Grade is B\n");
                        break;
                case 'C':
                        printf("Your Grade is C\n");
                        break;
                case 'D':
                        printf("Your Grade is D\n");
                        break;
                default:
                        printf("Your Grade is Low\n");
                        break;
        }
}
```

cases එකක් මගින් සිදු වනුයේ කිසියම් විචල්‍යය යම් අගයන් කිහිපයකට වරින් වර සමාන වේදැයි බැලීමයි. එසේ සමාන වේ නම් එය තුළ ඇති කේත ධාවනය වේ. මෙයත් බැලූ බැල්මය else if කේතයකට සමාන වේ යැයි සිතුනත් එසේ නොවේ. මෙහිදී එක් අවස්තාවක හෝ අදාළ සසඳන අගය හෝ අක්ෂරය සමාන වුවහොත් ඒ තුළ ඇති කේත ධාවනය වී break කේතයක් හමුවන තුරුම පහළට ගමන් කරයි. මෙම නිදසුනේ සියලුම අවස්ථා සඳහා break යොදා ඇති නිසා මෙහිදී සිදු වන්නේ else if එකක ආකාරයමයි.

8.2. Cases

8_cases_2.c

```
#include <stdio.h>

void main() {
        char x = 'A';

        switch(x) {
                case 'A':
                        printf("Your Grade is A\n");
                case 'B':
                        printf("Your Grade is B\n");
                case 'C':
                        printf("Your Grade is C\n");
                case 'D':
                        printf("Your Grade is D\n");
                default:
                        printf("Your Grade is Low\n");
        }
}
```

මෙම නිදසුන සැළකුවහොත් මෙහි කිසිම අවස්තාවක break නොමැති බැවින් x හි අගය අදාළ එකයම් හෝ case එකකට සමාන වූ විට එහි තුළ ඇති කේතය ක්‍රියාත්මක වී ඉන් ඉදිරියට ඇති ඉතිරි සියලුම cases තුළ ඇති කේතන ක්‍රියාත්මක වේ.

9.1. Loop While

9_loop_1_while.c

```
#include <stdio.h>

/*

        while(condition) {
                looping statements;
        }

*/

void main() {
        int i = 0;
        while(i < 10) {
                printf("i = %d\n", i);
                i++;
        }

        printf("\n");

        // 1 = true
        // 0 = false
        int x = 1;
        int n = 5;
        while(x) {
                printf("n = %d\n", n);
                n += 10;
                if(n>80) {x = 0;}
        }
}
```

යම් කිසි කේත කොටසක් කිහිප වරක් පුනරාවර්තනය කිරීමට Loops භාවිතා වේ. මෙහි දැක්වෙන්නේ While නම් වූ ඉන් එක වර්ගයකි. while() තුලට කිසියම් සැසඳීම් ප්‍රකාශයක් ලබා දිය යුතුය. එවිට එම ප්‍රකාශය සත්‍යවන තුරු while සමග ඇති {} සඟල වරහන් තුල ඇති කේත ක්‍රියාත්මක වේ. පළමු while loop එකෙහි සැසඳීම කර ඇත්තේ i යන්න 10ට අඩු වන තුරු ලෙසයි. ආරම්භයේදී x අගය 0 නිසා while කේතය ක්‍රියාත්මක වේ. while තුලදී x හි අගය එකින් එක වැඩි වන නිසා අවසානයේ x හි අගය 10 වූ විට while කේතය නවතී. ඒ වන තාක් i හි අගය පිලිවෙලින් පහළට දර්ශනය වේ. දෙවන නිදසුනේ ඇත්තේ x පදය සත්‍ය වන තුරු එම දෙවන while කෙතය ක්‍රියාත්මක වේ. while තුල ඇති if කේතය මගින් n හි අගය 80ට වඩා විශාල වූ විට x අගය 0 බවට පත් කරන අතර ඒ අවස්තාවේදී while කේතය නවතී.

9.2. Loop Do While

9_loop_2_do_while.c

```
#include <stdio.h>

/*

        do {
                looping statements;
        } while(condition);

*/

void main() {
        int i = 0;
        do {
                printf("i = %d\n", i);
                i++;
        } while(i < 10);
}
```

Do While Loop යන්නත් While Loop ආකාරයේම වන අතර එහි සැකැස්මද Main Function එකට ඉහළින් දක්වා ඇත.

9.3. Loop For

9_loop_3_for.c

```
#include <stdio.h>

/*

        for([start];[condition];[increasement]) {
                looping statements;
        }

*/

void main() {
        int i; // pre-declared variable
        for(i = 0; i < 10; i++) {
                printf("i = %d\n", i);
        }

        printf("Value of i after the loop = %d\n", i);

        printf("\n");

        // variable declaration for only the loop
        for(int r = 0; r < 10; r++) {
                printf("r = %d\n", r);
        }

        printf("\n");

        // for-loop with charactors
        for(char c = 'A'; c < 'K'; c++) {
                printf("c = %c\n", c);
        }
}
```

For Loop ආකාරය පෙර Loop ආකාර දෙකෙන්ම තරමක් වෙනස් වේ. මෙහිදී for() තුල ප්‍රධාන කොටස් තුනක් ලබා දිය යුතු අතර මුලින්ම අප භාවිත කරන විචල්‍යය‍ය් මුල් අවස්තාව, දෙවනුව සැසඳීම් පදය සහ තෙවනුව එක් වරක් කේත ක්‍රියාක්මක වූ පසු අදාල විචල්‍යයට කළ යුතු වෙනස්කම දක්වයි. පළමු for loop එකට පිටින් i විචල්‍යක් තනා ඇති අතර අපසුව for() තුල i හි අගය 0 කර තිබේ. පසුව එම i හි අගය 10ට අඩුවන තාක් වරින් වර i හි අගය එකින් එක වැඩි කරමින් ඒ තුළ ඇතු කේත ධාවනය කෙර්. අවසානයේ printf() එකක් මගින් i හි අගය දර්ශනය කර ඇත. මෙහිදී i හි අගය 10 ලෙස දැක්වෙනු ඇත. නමුත් දෙවන loop එකෙහි ඇති r විචල්‍යය භාවිත තළ හැකි වන්නේ එම loop එක තුළ පමණි. එම කේත ක්‍රියාත්මක වූ පසු r විචල්‍යය නැති වී යයි. තෙවන කේතයේ ඇත්තේ අක්ෂර මගින් for loop ලියා ඇති ආකාරයයි. නමුත් මෙහිද සිදු වන්නේ නිඛිල මගින් වන ක්‍රියාවලියක්ම වේ. මක් නිසාදයත් A යනු ASCII මගින් 65 වන අතර B, C, D යන අක්ෂර 66, 67, 68 ලෙස ගැනේ. මෙම loop එක මගින් පහළට A සිට J දක්වා ප්‍රතිදානය වේ.

10.1. Arrays Begin

10_arrays_1_begin.c

```
#include <stdio.h>

void main() {
	int l[10]; // array declaration with gabage values
	int g[5] = {45, 13, -12, 24, 63}; // array declaration with assiging values at same
time

	printf("l[%d] = %d\n", 0, l[0]); // printing the first item of array
	printf("l[%d] = %d\n", 1, l[1]);
	printf("l[%d] = %d\n", 2, l[2]);
	printf("l[%d] = %d\n", 3, l[3]);

	printf("\n");

	l[0] = 10; // assigning value to an item of the array
	printf("l[%d] = %d\n", 0, l[0]); // printing the first item of array

	printf("\n");

	printf("g[%d] = %d\n", 0, g[0]);
	printf("g[%d] = %d\n", 1, g[1]);
	printf("g[%d] = %d\n", 2, g[2]);
	printf("g[%d] = %d\n", 3, g[3]);
	printf("g[%d] = %d\n", 4, g[4]);
}
```

එක් විචල්‍යයක් තුල ගබඩා කල හැක්කේ එක් අගයක් පමණක් වේ. ඒ නමුත් ආරාවන් (Arrays) මගින් එකම දත්ත ආකාරයක අගයන් රාශියක් ගබඩා කළ හැකිය. මෙය සාමාන්‍ය විචල්‍ය නමක් ලෙසම දක්වා අගය කොට වරහන් [] යොදා හඳුන්වා දෙයි. මුලින්ම එම කොටු වරහන තුළ අදාල ආරාවේ විශාලත්වය නෙහොත් ගබා කළ හැකි දත්ත ගනන ලබා දේ. එය සුව වෙනස් කළ නොහැක. මෙම නිදසුනේ int l[10]මගින් නිඛිල දත්ත 10ක් ගබඩා කිරීමට ආරාවක් සෑදේ. මුල් අවස්තාවේ කිසිඳු අගයන් ලබා දීමක් කර නොමැති නිසා එම ආරාවේ සෑම අගයක්ම Garbage Value එකක් වේ. නමුත් g[5] ආරාව සෑදූ සැන්නිම් {} වරහන් මගින් කොමා සමග සියලු දත්ත ලබා දී ඇති නිසා එහි Gabage Value ඉතුරු නොවේ. එම ගබඩා කර දත්ත නැවත ලබා ගැනීමටද කොටු වරහන යෙදිය යුතුය. එහිදී l[0] යනු පළමු දත්තය වේ. l[1] යනු දෙවන දත්තය වේ. ඒ ආකාරයට දත්ත සංඛ්‍යාවට එකක් අඩු වන තෙක්ම දත්ත පවතී. ඒ ආකාරයට දත්ත කියවීම සහ දත්ත අලුත් කිරීම කළ හැකිය.

10.1. Arrays Display In For Loop

10_arrays_1_display_in_for_loop.c

```
#include <stdio.h>

void main() {
	int g[10] = {45, 13, 12, 24, 63, 53, 54, 73, 36, 73};

	for(int i = 0; i < 10; i++) {
		printf("g[%d] = %d\n", i, g[i]);
	}
}
```

For Loop එකක් මගින් ආරාවක අගයන් වරින් වර ප්‍රතිදානය මෙහිදී සිදු කර ඇත. i නම් විචල්‍යය 0 සිට 10ට අඩු වන තාක් For Loop තුල ක්‍රියාත්මක වන නිසා වරින් වර g[i] සැළකූ විට එක් දත්තයෙන් දත්තය ආරාවෙන් කියවිය හැකිය.

10.2. Arrays Assign In For Loop

10_arrays_2_assign_in_for_loop.c

```
#include <stdio.h>

void main() {
	int g[10];

	for(int i = 0; i < 10; i++) {
		printf("g[%d] = %d\n", i, g[i]);
	}

	printf("\n");

	for(int i = 0; i < 10; i++) {
		scanf("%d", &g[i]);
	}

	printf("\n");

	for(int i = 0; i < 10; i++) {
		printf("g[%d] = %d\n", i, g[i]);
	}
}
```

මෙහිදී ආරාවක සෑම අගයක්ම පිලිවෙලින් ආදානය කිරීමට for loop එකක් භාවිත කර තිබේ. මුලින්ම ඇති loop එක මගින් ආරාවේ Garbage Value සියල්ල ප්‍රතිදානය කරන අතර දත්ත ආදාරනය කර ඇත්තේ දෙවෙනි loop එක මගිනි. මෙහිදී ආරාවේ අදාළ ස්ථානයක ඇති දත්තය සැළකීමට g[i] ලෙස ගෙන ඇති අතර &g[i] මගින් එම දත්තය සඳහා RAM එකෙහි ගෙන ඇති Memory Block එකෙහි ලිපිනය ලැබේ. තුන් වන loop එක මගින් ආදානය කල පසු ආරාවේ අගයන් පවතින ආකාරය ප්‍රතිදානය කර ඇත.

10.3.1. Arrays Display As Table

10_arrays_3_display_as_table_1.c

```
#include <stdio.h>

void main() {
	int sid[8] = {196, 197, 198, 199, 200, 201, 202, 203};
	int age[8] = {22, 23, 21, 23, 22, 21, 20, 20};
	float avg[8] = {67.4, 65.92, 54.8, 94.73, 63.9, 65.35, 84.64, 70.0};

	printf("%5s %5s %8s\n", "ID", "Age", "Avarage");

	for(int i = 0; i < 8; i++) {
		printf("%5d %5d %8.2f\n", sid[i], age[i], avg[i]);
	}
}
```

මෙහිදී සිදු කර ඇත්තේ ආරාවන් තුනක දත්ත වගුවක් ලෙස ප්‍රතිදානය කර තිබීමයි. මෙවැනි අවස්තා වල printf() වල ඇති එකෙල්ල කිරීමේ ක්‍රමයේ වැදගත් කම වැටහෙනු ඇත.

10.4.2. Arrays Display As Table

10_arrays_4_display_as_table_2.c

```
#include <stdio.h>

void main() {
        int sid[5] = {196, 197, 198, 199, 200};
        int m_1[5] = {76, 84, 68, 37, 55};
        int m_2[5] = {97, 75, 74, 86, 68};

        printf("%8s %10s %10s %12s\n", "ID", "Marks_1", "Marks_2", "Avarage");

        for(int i = 0; i < 5; i++) {
                float avg = (m_1[i] + m_2[i]) / 2.0 ;
                printf("%8d %10d %10d %12.2f\n", sid[i], m_1[i], m_2[i], avg);
        }
}
```

මෙම නිදසුනේ ප්‍රතිදානය කර ඇති වගුවේ අවසාන තීරය තුළ විශයන් දෙකක සාමාන්‍ය ගණනය කර දක්වා ඇත. එය ගණනය කර ඇත්තේද වරින් වර for loop එක තුලය.