نیمسال اول 1402-03	عنوان: آشنایی با ساختارهای کنترلی و عملگر ها	اساتید: دکترمرادی، دکتر هاشمی

پیش از شروع، ابتدا ساختار کلی if را مرور می‌کنیم:

```
if (/* condition */)
    {
        /* code */
    }
```

که در آن ابتدا شرط یا condition بررسی گردیده و در صورت درست بودن آن، code اجرا می‌گردد.

توجه۱: در جایگاه condition ، هر عدد غیر صفر به معنی **true** و عدد صفر به معنی **false** می‌باشد.
توجه۲: گذاشتن "{" و "}" برای نوشتن قطعه کد مربوط به دستور if و else در برنامه‌هایی که بخواهیم تنها یک دستور در داخل if و else اجرا شود، اجباری نیست. ولی در حالتی که بخواهیم بیش از یک دستور در بخش if و یا else اجرا شوند، این مورد اجباری می‌شود.

۱- انجام دهید. (پیاده سازی دستورات شرطی به کمک if و else)

برنامه‌ی زیر را بخوانید. به نظر شما چه اتفاقی خواهد افتاد؟

```
#include<stdio.h>
int main() {
    int input;
    scanf("%d", &input);
    if (input = 5){
        printf("Your number was 5\n", input);
    }
    else if (input > 5) {
        printf("Your number was greater than 5\n");
    }
    else {
        printf("Your number was less than 5\n");
    }
    return 0;
}
```

حال آن را در محیط VScode نوشته و کامپایل و اجرا نمایید. آیا این برنامه همان طور که انتظار داشتید عمل می‌کند؟ چه اتفاقی افتاد؟

- در برنامه فوق، در انتهای برنامه با استفاده از دستور printf، مقدار خوانده شده از ورودی (input) را نمایش دهید. ببنید که چرا این برنامه مخالف انتظارتان پاسخ می‌دهد؟
- برنامه فوق را تصحیح کنید به صورتی که در ازای ورودی‌های مختلف خروجی مناسب را چاپ نماید.

قسمت ۱: علت را برای دستیاران آموزشی توضیح دهید.

۲- انجام دهید!(عملگر ها)

کد زیر را در یک پروژه جدید کامپایل و اجرا کنید.

```
#include<stdio.h>
int main(){
    int x=1 , y=0 , z=4;
    int a =x && y++;
    printf("x= %d\ny= %d\nz= %d\na= %d\n",x,y,z,a);
    return 0;
}
```

- چه اتفاقی افتاد؟ چرا؟

راهنمایی: با توجه به اینکه مقداردهی متغیر a توسط یک سری عبارات منطقی صورت گرفته، مقدار a، یا ۰ منطقی (false) و یا ۱ منطقی (true) خواهد بود.

- حال کد بالا را به کد زیر تغییر دهید.(تغییرات با هایلایت زرد مشخص شده اند.) چه تغییری رخ داد؟

```
#include<stdio.h>
int main(){
    int x=1 , y=0 , z=4;
    int a =x && ++y;
    printf("x= %d\ny= %d\nz= %d\na= %d\n", x,y,z,a);
    return 0;
}
```

- حال کد بالا را به کد زیر تغییر دهید.(تغییرات با هایلایت زرد مشخص شده اند.) چه تغییری رخ داد؟

```
#include<stdio.h>
int main(){
    int x=1 , y=0 , z=4;
```

```
    int a =--x && y || (z>>=2);
    printf("x= %d\ny= %d\nz= %d\na= %d\n",x,y,z,a);
    return 0;
}
```

راهنمایی: عبارت ***z >>= 2*** معادل عبارت ***z = z >> 2*** می‌باشد.

ابتدا کد زیر را بخوانید و حدس بزنید چه اتفاقی خواهد افتاد؟ سپس آنرا در یک پروژه جدید اجرا کنید.

```
#include<stdio.h>
int main(){
    int x=1 , y=0 , z=4;
    int a =--x && y || (z>>=2);
    !a ? printf("%d\n", z) : printf("%d\n", ++z);
    return 0;
}
```

قسمت ۲: علت بخش‌های بالا را برای دستیاران آموزشی توضیح دهید.

راهنمایی: عملگر[1] ***?*** یک روش مختصر برای نوشتن یک عبارت ***if-else*** می‌باشد. این عملگر سه قسمت دارد: یک شرط (***condition***)، یک مقدار برای بازگشت در صورتی که شرط درست باشد (***value_if_true***) و یک مقدار برای بازگشت در صورتی که شرط نادرست باشد (***value_if_false***). ساختار این عملگر به صورت زیر است:

Condition ? value_if_true : value_if_false

توجه: برای آشنایی بیشتر با عملگرها در زبان C، می‌توانید به این لینک مراجعه کنید.

پیش از انجام قسمت بعدی، ابتدا ساختار کلی حلقه‌های for و while را مرور می‌کنیم:

۱- ساختار کلی while:

```
while (/* condition */)
{
    /* code */
```

[1] Operator

```
}
```

که در هر مرحله، ابتدا شرط یا condition بررسی شده و در صورت درستی آن، code داخل حلقه اجرا می‌گردد.

همچنین اگر بخواهیم در ابتدا یکبار code اجرا شده و سپس شرط بررسی شود، می‌توان از ساختار زیر استفاده نمود:

```
do
{
    /* code */
} while (/* condition */);
```

۲- ساختار کلی for:

```
for (/*initialization*/; /* condition*/; /*update*/)
{
    /* code */
}
```

که در قسمت initialization، مقداردهی به متغیر صورت گرفته و در قسمت condition شرط اجرای حلقه (code) بیان شده و در قسمت update نیز تغییراتی که باید بعد از هر سری انجام حلقه روی متغیر اعمال شود، مشخص می‌گردد. در کل می‌توان گفت در هر حلقه‌ی for، بعد از آنکه مقدار دهی به متغیر صورت گرفت، ابتدا شرط بررسی می‌شود، سپس در صورت درست بودن آن، code اجرا می‌شود، و پس از اجرای آن، مقدار متغیر بروزرسانی می‌گردد. سپس دوباره شرط بررسی شده و ...

همچنین هر حلقه‌ی for را می‌توان با یک حلقه‌ی while پیاده سازی کرد و برعکس. به عبارت دیگر قطعه کدهای زیر معادل یکدیگر هستند:

`for (/*initialization*/; /* condition*/; /*update*/)`
`{`
`  /* code */`
`}`
	`/*initialization*/;`
	`while (/* condition */)`
	`{`
	`  /* code */`
	`  /*update*/`
	`}`

۳- انجام دهید!

حال قصد داریم با استفاده از حلقه‌ی for، برنامه‌ای بنویسیم که زوج یا فرد بودن اعداد را مشخص می‌کند.
برنامه‌ی زیر را که بدین منظور نوشته شده است در یک پروژه‌ی جدید کامپایل کنید.

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

```
    for(int i=1;i<=10;i--){
        printf("%d ",i);
        if(i%2==0){
            printf("is even!\n");
        }else{
            printf("is odd!\n");
        }
    }
    return 0;
}
```

- به نظرتان این برنامه چه ایرادی دارد؟ آیا در هنگام کامپایل، با پیام خطا مواجه شدید؟ حال این برنامه را اجرا کنید. چه اتفاقی افتاد؟ علت را برای دستیاران آموزشی توضیح داده و برنامه را اصلاح نمایید.(قسمت ۳)

"**توجه:** در VScode، برای متوقف کردن هر برنامه‌ای بعد از اجرا کافیست کلیدهای ترکیبی **ctrl+C** را بفشارید."

- حال یکبار شرط حلقه (condition) و بار دیگر قسمت بروزرسانی آن (update) را حذف کرده و سپس برنامه را اجرا کنید. علت اتفاق پیش‌آمده در هر بار را برای دستیاران آموزشی توضیح دهید.(قسمت ۴)

توجه: در حلقه‌ی for می‌توان هر یک از قسمت‌های شرط و یا بروزرسانی را حذف کرد ولی در هر صورت، باید هردو علامت **;** در حلقه حضور داشته باشند.

٤- انجام دهید!

در این قسمت می‌خواهیم برنامه‌ای بنویسیم که در هر مرحله یک عدد را از ورودی دریافت کند و تا زمانی که عدد صفر وارد نشده، به کار خود ادامه دهد. پس از وارد شدن عدد صفر، این برنامه باید جمع اعداد وارد شده را چاپ کند.
کد زیر بدین منظور نوشته شده است:

```
#include <stdio.h>
int main(){
    int num;
    int result=...;
```

```
    scanf("%d", &num);
    while(...){
        result=result+num;
        scanf("%d", &num);
    }
    printf("%d\n", result);
    return 0;
}
```

- در کد بالا، جاهای خالی را که هایلایت نیز شده‌اند، با عبارات مناسب تکمیل کنید. نتیجه را به دستیاران آموزشی نشان دهید.(قسمت۵)
- عبارت `result=result+num` را با چه عبارتی می‌توان جایگزین کرد؟ جهت راهنمایی به این لینک مراجعه کنید.
- حال همین قطعه کد را با ساختار do-while پیاده سازی کنید. نتیجه را به دستیاران آموزشی نشان دهید.(قسمت۶)

۵- انجام دهید!(حلقه‌ی بی‌نهایت!)

در این قسمت می‌خواهیم حلقه‌ای را پیاده سازی کنیم که هیچ گاه به پایان نرسیده و مدام در حال تکرار شدن باشد! اصطلاحا به این حلقه‌ها، حلقه‌ی بی‌نهایت[2] گفته می‌شود. به برنامه‌ی زیر توجه کنید:

```
#include <stdio.h>
int main(){
    int n;
    int value =1;
    while (value){
        scanf("%d",&n);
        printf("%d\n",n);
    }
}
```

همانطور که می‌بینید، این برنامه هر بار یک عدد از ورودی گرفته و سپس همان را در خروجی نمایش می‌دهد و این کار را تا زمانی که برنامه را متوقف نکرده‌اید تکرار می‌کند.

- حال اگر بخواهیم برنامه‌ای را بنویسیم که همین‌کار را انجام دهد ولی در شرایط خاصی مثل وارد کردن عدد ۰، متوقف شود، چکار باید کرد؟ نتیجه را به دستیاران آموزشی نشان دهید.(قسمت۷)

راهنمایی ۱: یک راه برای خروج از حلقه، آن است که شرط موجود در while، نادرست یا ۰ گردد.

راهنمایی ۲: می‌توانید از قطعه کد زیر کمک بگیرید!

```
    int value =1;
    while (value){
        scanf("%d",&n);
        if (...){
            value = ...;
        }else{
```

[2] Infinite loop

```
        . . .
    }
}
```

٦- انجام دهید!(حلقه‌های تو در تو!)

قطعه کد زیر مربوط به برنامه‌ای است که ابتدا عدد a را از ورودی گرفته و سپس مربعی a در a، با کارکترهای # را در خروجی نمایش می‌دهد.

```
#include<stdio.h>
int main(){
    int a;
    printf("please enter a number: ");
    scanf("%d", &a);
    for(int i=1 ; i<=a ; i++){
        for(int j=1 ; j<=a ; j++){
            printf("#");
        }
        printf("\n");
    }
    return 0;
}
```

- پس از کامپایل کد و بررسی آن، روند انجام برنامه در داخل حلقه‌های تو در تو و همچنین دلیل وجود عبارت هایلایت شده در آن قسمت برنامه را به دستیاران آموزشی توضیح دهید.(قسمت۸)
- حال برنامه‌ی بالا را طوری تغییر دهید تا ۲ عدد a و b را از ورودی گرفته و سپس مستطیلی با ابعاد a در b با کاراکترهای # را در خروجی نمایش دهد. نتیجه را به دستیاران آموزشی نشان دهید.(قسمت۹)
- اکنون برنامه‌ی قسمت قبل را به نحوی تغییر دهید که اعداد a و b را از ورودی خوانده و جدول ضربی به اندازه‌ی a سطر و b ستون را در خروجی چاپ کند. برای مثال خروجی کد شما به ازای a=5 و b=6 باید به صورت زیر باشد:

```
1 2 3 4 5 6
2 4 6 8 10 12
3 6 9 12 15 18
4 8 12 16 20 24
5 10 15 20 25 30
```

قسمت ۱۰: حال نتیجه را به دستیاران آموزشی نشان دهید.

۷- انجام دهید!(دستورات کنترل غیر شرطی)

ابتدا عملکرد دستوارت break و continue را مرور می‌کنیم:

دستور break: این دستور موجب خروج از داخلی‌ترین حلقه‌ی تکرار می‌شود.

دستور continue: این دستور در حلقه‌ی تکرار موجب انتقال کنترل به ابتدای حلقه می‌شود. سپس شرط حلقه مورد بررسی قرار می‌گیرد، چنانچه شرط درست باشد، اجرای دستورات حلقه ادامه می‌یابد وگرنه حلقه‌ی تکرار خاتمه می‌یابد.

حال می‌خواهیم برنامه‌ای بنویسیم که ۵ عدد را از کاربر دریافت کرده و بدون در نظر گرفتن مقادیر منفی، حاصل جمع آن‌ها را در خروجی نمایش دهد. یک پروژه‌ی جدید اجرا کرده و برنامه‌ی زیر را کامپایل و اجرا کنید.

```
#include <stdio.h>
int main(){
    int n , sum=0;
    for(int i=1;i<=5;i++){
        printf("Enter a n%d: ", i);
        scanf("%d", &n);
        if(n>=0){
            sum +=n;
        }
    }
    printf("Sum= %d\n", sum);
    return 0;
}
```

- به نظرتان برنامه‌ی بالا را با کدام یک از دستورات break یا continue می‌توان پیاده سازی کرد؟ این کار را انجام دهید و سپس نتیجه را به دستیاران آموزشی نشان دهید.(قسمت ۱۱)
- همانطور که دیدید، یک راه دیگر برای خروج از حلقه، استفاده از دستور break می‌باشد. حال برنامه‌ی قسمت ۷ را یکبار دیگر با استفاده از این دستور، پیاده سازی کرده و نتیجه را به دستیاران آموزشی نشان دهید.(قسمت ۱۲)

پیش از انجام قسمت بعدی، ابتدا ساختار کلی switch را مرور می‌کنیم:

```
switch (expression)    {
    case /* <مقدار ۱> */:
        /* code 1 */
        break;
    case /* <مقدار ۲> */:
        /* code 2 */
        break;
    default:
        /* code*/
        break;
    }
```

ابتدا مقدار عبارت موجود در مقابل switch یعنی expression تعیین می‌گردد. اگر این مقدار با <مقدار ۱> برابر بود، code 1 اجرا می‌شود و سپس دستور break کنترل برنامه را از ساختار switch خارج می‌کند. ولی اگر این مقدار با <مقدار ۱> برابر نبود، با <مقدار ۲> مقایسه می‌شود و اگر با <مقدار ۲> برابر بود، code 2 اجرا می‌شود و سپس دستور break کنترل برنامه را از ساختار switch خارج می‌کند. تا زمانی که عبارت

محاسبه شده (expression) با یکی ازمقادیر ذکر شده برابر نبود، عمل مقایسه با مقدار بعدی ادامه می‌یابد. در صورتی که مقدار آن با هیچ یک از مقادیر case ها برابر نبود، دستورات قسمت default اجرا شده و کنترل از ساختار switch خارج می‌شود. توجه کنید که قسمت default می‌تواند فقط شامل break باشد.

۸- انجام دهید!

در این قسمت می‌خواهیم برنامه‌ای بنویسیم که ابتدا شماره‌ی روز هفته (عددی بین ۱ و ۷) را از ورودی خوانده و سپس با استفاده از ساختار switch، نام روز معادل با آن را در خروجب نمایش دهد. به برنامه‌ی زیر توجه کنید:

```
int main(){
    int n;
    scanf("%d",&n);
    switch (n){
    case 1:
        printf("Shanbeh\n");
        break;
    case 2:
        printf("Yekshanbeh\n");
        break;
    case 3:
        ...
    default:
        break;
    }
    return 0;}
```

- ابتدا ادامه‌ی برنامه (قسمتی که با هایلایت مشخص شده است) را تکمیل نموده و سپس برنامه را به نحوی تغییر دهید که در صورت وارد کردن مقداری نادرست (هر مقادیری به‌جز اعداد ۱ تا ۷)، پیام "Invalid input!" در خروجی نمایش داده شود. (دقت کنید که مجاز به استفاده از if نمی‌باشید!). نتیجه را به دستیاران آموزشی نشان دهید.(قسمت ۱۳)
- حال در case 3، عبارت break را حذف کنید و دوباره برنامه را اجرا کرده و عدد ۳ را به عنوان ورودی وارد نمایید. چه اتفاقی افتاد؟ اکنون در دو case 3 و case 4 ، عبارات break را حذف کرده و برنامه را دوباره با ورودی ۳ اجرا کنید. چه نتیجه‌ای می‌گیرید؟ علت را برای دستیارن آموزشی توضیح دهید.(قسمت ۱۴)

۹- بازی حدس عدد(امتیازی!)

با استفاده از ساختار if و else که در قسمت‌های پیشین فرا گرفتید، می‌خواهیم تا در این قسمت با اقتباس از الگوریتم مشهوری به نام جستجوی دودویی (binary search) عددی تصادفی را که بین ۱ تا ۱۰۰ انتخاب شده است، پیدا کنیم. روش تولید یک عدد تصادفی بین ۱ تا ۱۰۰ در زبان c به شکل زیر است که در آن، متغیّر

seedرا با شماره‌ی دانشجویی خود مقدار بدهید. (شما می‌توانید هر مقدار دلخواهی را به متغیّر تعریف شده بدهید. اینجا به دلیل اینکه نتیجه هر فرد حتی الامکان متفاوت بشود، بهتر است که هر دانشجو شماره‌ی دانشجویی خودش را به عوان مقدار به این متغیر بدهد.)

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

int main(){
    int seed = 810102000;
    srand(seed);
    int random_number = rand() % 100;
      ...
```

در ادامه، در یک حلقه‌ی while(true) ، ساختارهای کنترلی‌ای پیاده سازی کنید که رفتارهای زیر را داشته باشد:

- ابتدا از کاربر بخواهد که یک عدد بین ۱ تا ۱۰۰ حدس بزند.
- چنانچه عدد وارد شده برابر با عدد تصادفی تولید شده بود، حلقه‌ی while(true) به اتمام برسد.
- چنانچه عدد وارد شده از عدد تصادفی تولید شده کوچک‌تر بود، پیامی کوتاه به کاربر بدهد که عدد او از عدد تصادفی کوچک‌تر است.
- در صورت بزرگ‌تر بودن عدد وارد شده از عدد تصادفی تولید شده، پیامی دیگر داده شود تا کاربر از این موضوع آگاه گردد.

چگونه کاربر می‌تواند در سریع‌ترین زمان عدد تصادفی تولید شده را حدس بزند؟

برای آشنایی بیشتر با جستجوی دودویی، می‌توانید به این لینک و یا این لینک و یا سایت‌های مشابه مراجعه کنید.

قسمت ۱۵: حال نتیجه را به دستیاران آموزشی نشان دهید.

توضیحات بیشتر : تولید اعداد تصادفی از مسائل مهم حوزه‌ی علوم کامپیوتر است؛ و مهم‌ترین کاربرد آنها در حوزه‌ی امنیت شبکه به چشم می‌خورد. از آنجا که در کامپیوتر تمام کارها باید دقیقا مشخص باشند، لذا مجبوریم تا به جای انجام کارهایی مانند پرتاب سکه برای تولید کاملا تصادفی هر بیت از یک رشته دودویی، از توابعی استفاده کنیم که رفتارهایی شبه تصادفی دارند. چنانچه از تعریف یک تابع مشخص است، لازم است تا ضابطه‌ای داشته باشد و داشتن چنین خاصیتی، یعنی تولید اعداد به صورت کاملا تصادفی صورت نمی‌گیرد. چراکه در هر حال پس از گذشت زمان و تولید اعداد شبه تصادفی، به دلیل داشتن حالت‌های محدود (هر چقدر هم زیاد باشند، بی‌نهایت نخواهند بود) تابع تولید کننده آنها شروع به تولید اعداد تکراری خواهد کرد. به عبارت دیگر، هر تابع تولید کننده اعداد شبه تصادفی، یک دوره‌ی تناوب دارد. لذا یک راهکار برای این مشکل، افزایش دوره‌ی تناوب تابع تولید کننده‌ی اعداد تصادفی است. در کد شما، تابعی از زبان c استفاده شده است که با کمک گرفتن از عددی به نام seed در محاسبات خود، رشته‌ای از اعداد تصادفی را تولید می‌کند. حال اگر تعداد بیت‌های seed زیاد باشد، جستجو در فضای آن مشکل‌تر خواهد بود و لذا نمی‌توان با دانستن ضابطه‌ی تابع تولید کننده‌ی اعداد شبه

تصادفی، seed ای را یافت که رشته‌ای از اعداد را تولید کند که نهایتا به 1 عدد شبه تصادفی در دست ما برسد.[۳]

موفق باشید.

تهیه و تنظیم: امیرمرتضی رضائی

[3] برگرفته از اسلایدهای درس امنیت شبکه جناب آقای دکتر بهنام بهرک، پاییز ۹۶