



بِنام خدا

زبان برنامه نویسی C (21814)

# Lecture 1

## Chapters 1&2

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814)

نام درس:

زبان برنامه نویسی C (21814)

دلیل انتخاب درس؟

معرفی درس:

- هدف ارائه مطالب به صورت کاربردی
- برنامه نویسی به عنوان ابزار یک مهندس یا محقق
- نامحدود بودن علم و امکان پذیر نبودن داشتن همه آن
- فرض: آشنایی دانشجویها با يك زبان برنامه نویسی

## زبان برنامه نویسی C (21814)

مدرس:

• ناصر سلماسی

— تلفن 5738 – 6616

— آدرس الکترونیکی (Email): [nsalmasi@sharif.edu](mailto:nsalmasi@sharif.edu)

— ساعت های رفع اشکال:

○ یکشنبه ها و سه شنبه ها از ساعت 13:30 الي 15:00

## زبان برنامه نویسی C (21814)

**مربی های حل تمرین:**

• **مصطفی افخمی زاده**

– آدرس الکترونیکی: [mostafa.afkhamizadeh@gmail.com](mailto:mostafa.afkhamizadeh@gmail.com)

– ساعت های رفع اشکال:

• چهارشنبه ها 11 الی 12 صبح

• **بهزاد نعمتی**

– آدرس الکترونیکی: [bz\\_nemati@yahoo.com](mailto:bz_nemati@yahoo.com)

– ساعت های رفع اشکال:

• دوشنبه ها 12:30 الی 14:00

## زبان برنامه نویسی C (21814)

### نحوه مطالعه درس در طول ترم

- 4 تا 6 ساعت کار در هفته (مطالعه و برنامه نویسی)
- حضور در کلاس و مشارکت در مباحث
- تمرین ها و یا پروژه های هفتگی
- کوییز های هفتگی (ساده ولی موثر)
- تمرین های کلاسی در داخل کلاس

## زبان برنامه نویسی C (21814)

### مراجع:

- **Fisher A. E., Eggert D.W., and Ross, S.M., 2001, Applied C: an Introduction and More, McGraw-Hill.**
- هر کتابی مربوط به برنامه نویسی به زبان C

## زبان برنامه نویسی C (21814)

نمره بندي:

نمره	شرح	نمره	شرح
6	پروژه	2.5	تمرین
4	میان ترم	2.5	کوییزها
5	پایان ترم		

## زبان برنامه نویسی C (21814)

نمره بندی (ادامه):

تمرین ها:

- تمرین های هفتگی برای برنامه نویسی
- ارسال تمرین ها قبل از ساعت کلاس در روز موعود تحویل به  
آدرس [ie\\_21814@yahoo.com](mailto:ie_21814@yahoo.com) (عدم پذیرش در صورت  
تاخیر)

کویزها:

- هر هفته یک کویز در مورد مطلب های گفته شده در هفته قبلی



## زبان برنامه نویسی C (21814)

نمره بندي (ادامه):

پروژه ها:

- چهار پروژه (مهمترین قسمت درس)
- امکان تحویل یکی از پروژه ها با سه روز تاخیر
- کسر روزانه 5% نمره بابت تاخیر اضافی
- رایه گزارش کامل در پروژه ها
- تایپ گزارش

## زبان برنامه نویسی C (21814)

### نمره بندي (ادامه):

- رایه گزارش بصورت کتبی به همراه ارسال فایل برنامه بصورت الکترونیکی (می توان هر دو را بصورت EMAIL ارسال نمود).
- پروژه ها و تمرین ها بصورت انفرادی هستند. مشورت با سایرین تشویق می شود.
- در صورت اشکال در ساعت های رفع اشکال مراجعه و یا با ارسال EMAIL سوال کنید.

## زبان برنامه نویسی C (21814)

### نمره بندي (ادامه):

- تمرین ها، پروژه ها و مسایل مهم از طریق وب سایت کلاس اطلاع رسانی می شود.

<http://courseware.sharif.edu>

- دانشجویان موظف به رجوع مرتب به وب سایت درس هستند.

- فایل های برنامه را بصورت زیر نامگذاری کنید:

**Yourname\_HW3.doc      Yourname\_HW3.cpp**  
**Yourname\_Project3.doc    Yourname\_Project3.cpp**

## زبان برنامه نویسی C (21814)

### توصیه های مهم:

- در این درس تنها روش یاد گیری تمرین است. لذا تمرین های زیادی ارائه خواهد شد.
- انتظار دریافت پاسخ در مورد هر سوالی را نداشته باشید.
- آموزش در کلاس. سعی کنید کلیه مطالب را در کلاس متوجه شوید و پس از کلاس پیاده کنید (در مورد مثال ها).

## زبان برنامه نویسی C (21814)

### چرا زبان C؟

- مزایای زبان C :
  - نسبت به سایر زبان های برنامه نویسی بسیار سریع و فشرده
  - بی نهایت قابل انعطاف
  - مقدمه بسیار خوبی برای یادگیری زبان هایی مانند C# , JAVA, JAVA SCRIPT و C++ و غیره

## زبان برنامه نویسی C (21814)

### فصل اول:

### کامپیوتر و سیستم بر عهده دانشجو

## زبان برنامه نویسی C (21814)

### فصل دوم:

### برنامه و برنامه نویسی

## زبان برنامه نویسی C (21814) - فصل دوم

مزایای برنامه نویسی و استفاده از کامپیوتر:

انجام کارهای تکراری بصورت دقیق با سرعت فراوان

مثال:

محاسبه حقوق کارمندان یک شرکت



## زبان برنامه نویسی C (21814) - فصل دوم

چه کارهایی را

نمی توان با برنامه نویسی انجام داد؟

## زبان برنامه نویسی C (21814) - فصل دوم

### تعاریف:

- **برنامه:**

یک برنامه برای کامپیوتر مجموعه ای از دستورالعمل هاست. یک برنامه، بر حسب دستورالعمل ها داده ها را دریافت، پردازش های مورد نیاز را انجام و نتایج را ارائه می دهد. (نظیر سیستم پرداخت حقوق).

- بصورت دقیق تر، کامپیوتر یک برنامه را با اجرای دستورالعمل هایش به روی داده ها اجرا می کند.

## زبان برنامه نویسی C (21814) - فصل دوم

### تعاریف (ادامه):

- کامپیوتر به خودی خود تنها قادر به درک دستورات ساده نظیر جمع نمودن دو عدد است. لذا برای انجام یک عمل ساده نیاز به اجرای دستور العمل های طولانی است.
- برنامه نویسی یک کار غیر طبیعی است. عموماً انسانها مسایل خود را حل می کنند بدون اینکه بدانند چگونه این کار را انجام می دهند.
- انسانها عملاً می توانند وظایف خود را به کامپیوتر منتقل کنند اگر بتوانند مراحل اجرای هر کار را دقیقاً (قدم به قدم) به کامپیوتر منتقل نمایند (الگوریتم).

## زبان برنامه نویسی C (21814) - فصل دوم

### تعاریف (ادامه):

- **الگوریتم:**

الگوریتم یک روش حل برای یک مساله ساختار یافته است. این روش بایستی:

- بصورت کامل تعریف شده باشد.
- شفاف باشد.
- موثر باشد.
- بصورت مکانیکی قابل اجرا باشد.
- خاتمه داشته باشد.

## زبان برنامه نویسی C (21814) - فصل دوم

تعاریف (ادامه):

### • PSEUDOCODE

نوعی الگوریتم شفاهی است که الگوریتم را به کمک زبان نوشتاری و زبان برنامه نویسی شرح می دهد.

## زبان برنامه نویسی C (21814) - فصل دوم

وظیفه شما به عنوان یک دانشجو:

- یاد گیری زبان برنامه نویسی به عنوان یک ابزار
- یادگیری ایجاد الگوریتم برای حل مسائل

## زبان برنامه نویسی C (21814) - فصل دوم

- **مسائل مهم در برنامه نویسی**

( با یک مثال: نوشتن الگوریتمی برای یک سری هندسی)

$1, 5, 25, \dots pq^n$

- **تعریف دقیق مساله**
- **تعریف هدف مساله**
- **معرفی ورودی ها و خروجی ها**
- **معرفی ثابت ها و فرمول ها**
- **معرفی اصول محاسباتی مورد نیاز (نوع، محدوده، و دقت متغیرها)**

## زبان برنامه نویسی C (21814) - فصل دوم

### تمرین:

مشخصات اصلی یک برنامه را که واحد دمای فارنهایت را به سانتی گراد تبدیل می کند، بیان کنید:

$$F = 1.4C + 32$$



## زبان برنامه نویسی C (21814) - فصل دوم

### تایید و تصدیق (Verification):

برای آزمایش صحت برنامه ایجاد شده می توان برنامه را با حل  
چندین مساله ساده آزمایش نمود.

## زبان برنامه نویسی C (21814) - فصل دوم

### طراحی و ایجاد پاسخ:

- هنگامی که برنامه مورد تایید قرار گرفت، 3 نکته زیر باید مورد توجه قرار گیرد:

– الگوریتم

– روند اجرای برنامه

– آزمایش و غلط یابی

## زبان برنامه نویسی C (21814)- فصل دوم

### برنامه نویسی قدیمی

به علت کمبود وقت کامپیوتر (کارت پانچ)

## زبان برنامه نویسی C (21814) - فصل دوم

### نحوه برنامه نویسی:

ایجاد یک پوشه برای هر برنامه

### نحوه ایجاد فایل اجرایی برنامه:

- برنامه C++ را اجرا کنید.
- از منوی بالا **File/New** را انتخاب کنید.
- سپس منوی جدیدی پدیدار می شود.
- در قسمت فایل **C++ Source file** را انتخاب کنید.
- **File Name and Location** را انتخاب نمایید.
- کامپایل کردن و اجرا کردن



بِنام خدا

زبان برنامه نویسی C (21814)

Lecture 2

Chapter 3

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814)

### فصل سوم: مفاهیم اساسی

- زبان C یک زبان برنامه نویسی بسیار قوی
- مورد استفاده برای حل بسیاری از مسایل پیچیده جهان
- امکان برنامه نویسی با یادگیری تعدادی از مفاهیم اساسی

## زبان برنامه نویسی C (21814) - فصل سوم

### قسمتهای اصلی یک برنامه:

یک برنامه به زبان C شامل

- دستورات اجرایی
- یادداشت های برنامه نویس (comments)
- تعاریف اجزای برنامه
- تعاریف توابع به کار رفته در برنامه

## زبان برنامه نویسی C (21814) - فصل سوم

قسمتهای اصلی یک برنامه:

- **Comments :**
  - `/* */`
  - `//`
- **Preprocessor commands**
- **Words**
- **Declaration**
- **Statements**



## زبان برنامه نویسی C (21814) - فصل سوم

### Preprocessor commands

- دستوراتی که در ابتدای برنامه معرفی و اطلاعات مورد نیاز کامپایلر و مکان آنها را برای پردازش قسمت های مختلف برنامه فراهم می کنند.
- آغاز تمامی preprocessor command ها با علامت #

## زبان برنامه نویسی C (21814) - فصل سوم

### Words

- تعریف بسیاری از کلماتی که برای نوشتن یک برنامه به کار می روند، توسط زبان استاندارد برنامه نویسی
- این کلمات به عنوان کلمات کلیدی زبان برنامه نویسی C
- فهرست این کلمه ها در Appendix C کتاب (در وب سایت)
- تعریف سایر کلمات و دستورات مورد نیاز برنامه نویسی توسط کاربر

## زبان برنامه نویسی C (21814) - فصل سوم

### Declaration

- وظیفه **declaration** ها تعریف یک لغت جدید برای مجموعه کلمه های قابل درک توسط برنامه است.
- هر کلمه ای که در برنامه به کار می رود و متعلق به کلمه های قابل فهم زبان C نباشد، باید توسط کاربر قبل از استفاده در برنامه تعریف شود.

## زبان برنامه نویسی C (21814) - فصل سوم

### Statements

- یک **statement** همانند یک جمله است و عملی را انجام می دهد.

## زبان برنامه نویسی C (21814) - فصل سوم

### یک برنامه بسیار ساده:

```
/* This is a very simple Program*/    // Comment
#include <stdio.h>                      // Preprocessor Commands
                                       // input/output Standard
void main (void)                      // header line (Function)
{
    puts(" Salam");                   //
    printf(" Hello\n");               // block of codes
}
```

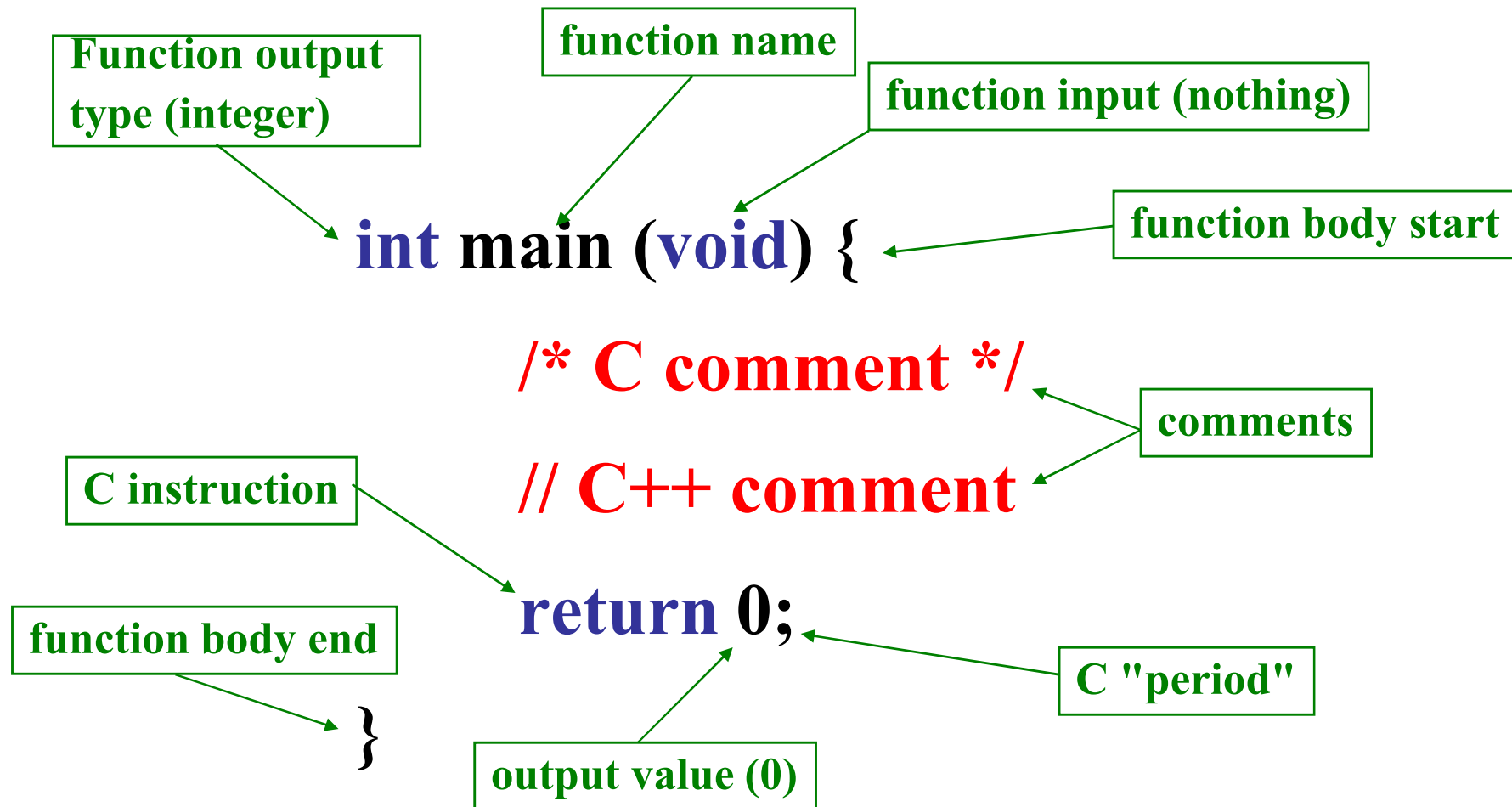
## زبان برنامه نویسی C (21814) - فصل سوم

### Function ها:

- یک برنامه به زبان C شامل تعدادی **Function** است. عملاً **Function** ها نقطه آغاز اجرای برنامه هستند.
- هر برنامه به زبان C با یک **Function** به نام **main** آغاز می شود.
- **Function** ها داده های ورودی را گرفته، پس از پردازش خروجی ایجاد می کنند.

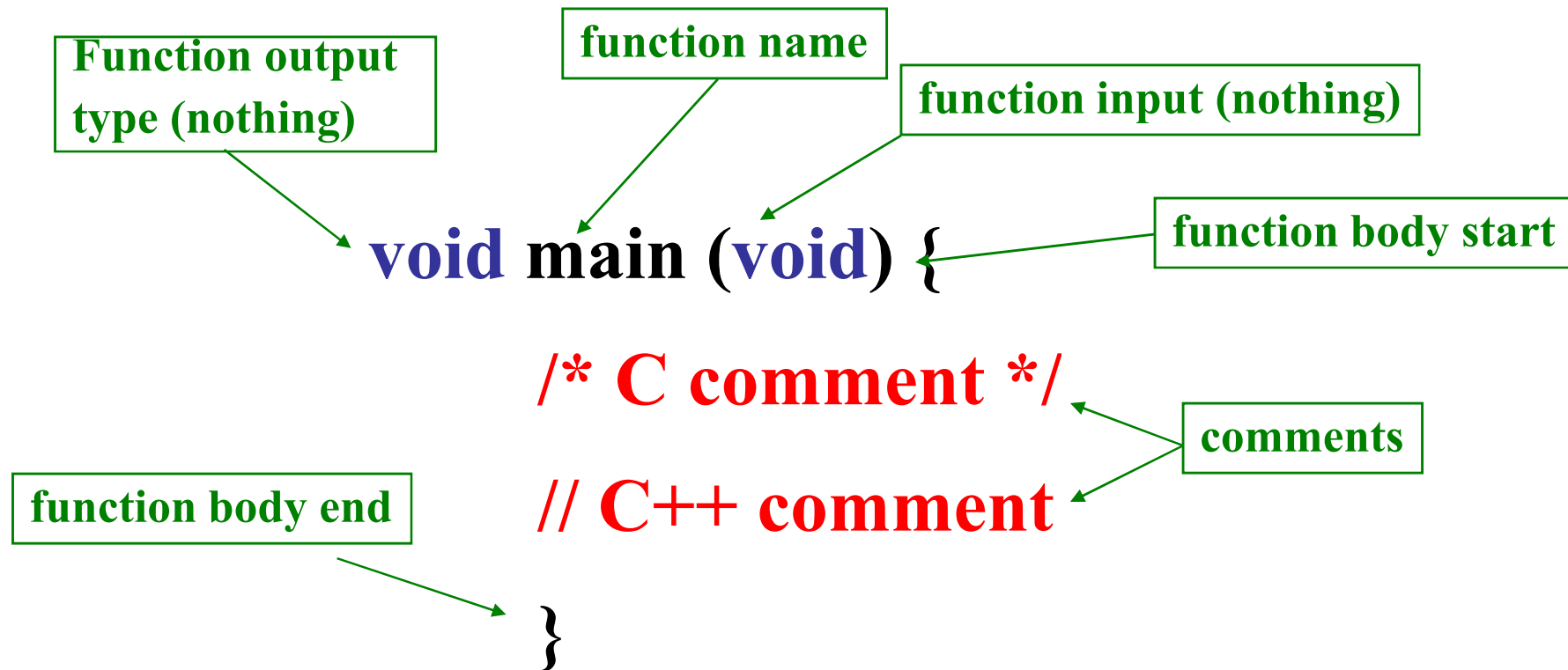
## زبان برنامه نویسی C (21814) - فصل سوم

### Function Structure



## زبان برنامه نویسی C (21814) - فصل سوم

### Function Structure





## زبان برنامه نویسی C (21814) - فصل سوم

### Function ها:

- زبان C شامل تعداد زیادی Function آماده است.
- در اغلب برنامه ها کاربر ملزم به نوشتن Function هایی برای انجام کارهای مورد نظر خود است.

## زبان برنامه نویسی C (21814) - فصل سوم

**#include <stdio.h>**

- **Standard I/O library**

## زبان برنامه نویسی C (21814) - فصل سوم

### **void main (void)**

- **Standard I/O library**

- گاهی توابع هیچ مقداری را در جایی که صدا زده می شوند بر نمی گردانند. معمولاً هدف از بکار بردن چنین توابعی ایجاد تأثیرات جانبی مانند تغییر دادن مقدار یک متغیر می باشد. به چنین توابعی توابع void گفته می شود زیرا در ابتدای آنها از کلمه void استفاده می شود.

## زبان برنامه نویسی C (21814) - فصل سوم

- **Strings** کلماتی هستند که میان " " ها قرار می گیرند.

## زبان برنامه نویسی C (21814) - فصل سوم

### متغیر ها، ورودی ها، خروجی ها و ترتیب آنها:

- همانطور که بیان شد، هدف از اجرای یک برنامه، انجام پردازش هایی به روی داده ها است.
- پیش از آغاز نوشتن یک برنامه بایستی بدانیم که اطلاعات مورد نیاز را از کجا و چگونه بدست آوریم، چگونه آنها را ذخیره و استفاده کنیم و چگونه پاسخ ها را گزارش دهیم.

## زبان برنامه نویسی C (21814) - فصل سوم

### متغیر ها (Variables):

- یک متغیر قسمتی از حافظه کامپیوتر است که توسط برنامه به آن نامی داده شده و برای ذخیره سازی اطلاعات و بازیابی آن بکار می رود.
- متغیر می تواند هر نامی داشته باشد به جز نام دستورات قابل درک زبان C.
- یکسری دستورالعمل ها برای انتخاب نام متغیر وجود دارد.
- برنامه نویس متبحر سعی در انتخاب نام مناسب و کوتاه دارد.

## زبان برنامه نویسی C (21814) - فصل سوم

### متغیر ها (ادامه):

- هر متغیر دارای نوع (Type) می باشد (عددی یا حرفی). میزان فضای اختصاص یافته برای هر متغیر بستگی به نوع آن دارد.
- هر متغیر با **Declaration** تعریف میشود. متغیر ها معمولاً در ابتدای برنامه تعریف می شوند. بهتر است پس از **Preprocessor Commands** ها تعریف شوند.

```
int Minute;  
double Second;
```

## زبان برنامه نویسی C (21814) - فصل سوم

### متغیر ها (ادامه):

- برای تعریف هر متغیری کامپایلر فضایی را برای متغیر تخصیص می دهد. این فضای اختصاص یافته به نام متغیر مرتبط می شود.
- گاهی اوقات در برنامه نویسی مجبور به رجوع به آدرس متغیر می شویم که در چنین حالتی از **ampersand** در جلوی نام متغیر استفاده می کنیم. **&Minute**



## زبان برنامه نویسی C (21814) - فصل سوم

نحوه اختصاص مقادیر به متغیر ها:

```
int age;      //declaration of integer variable age
```

```
age = 20;     //assignment of value 20
```

**or**

```
int age = 20; //declaration & assignment
```

## زبان برنامه نویسی C (21814) - فصل سوم

### ورودی ها و خروجی ها:

فرمت های ورودی و خروجی جزو پیچیده ترین قسمت های برنامه نویسی C می باشند. ساده ترین آنها عبارتند از:

کارکرد	مفهوم	نام دستور
نوشتن پیام به روی خروجی	نمایش string	<code>puts( )</code>
نوشتن پیام و اطلاعات به روی خروجی	چاپ خروجی با بکار گیری فرمت	<code>printf( )</code>
دریافت اطلاعات از صفحه کلید	دریافت ورودی با بکار گیری فرمت	<code>scanf( )</code>

## زبان برنامه نویسی C (21814) - فصل سوم

- این توابع همگی در **stdio library** تعریف شده اند که با نوشتن آن در ابتدای برنامه به برنامه اضافه شده اند.

- تفاوت میان **puts** و **printf**

- بکارگیری فرمت های ورودی:

```
scanf ("%i", &minutes);
```

```
scanf ("%i %i", &age, &weight);
```

## زبان برنامه نویسی C (21814) - فصل سوم

بکارگیری فرمت های خروجی:

- معمولاً فرمت های خروجی پیچیده تر هستند.

**Age = 21;**

**Weight = 65;**

**printf("Age: %i Weight: %i \n", Age, weight);**

**Age: 21 Weight: 65**

## زبان برنامه نویسی C (21814) - فصل سوم

بکارگیری فرمت های خروجی:

• \t

• \n

• \n\n\n

• " "

## زبان برنامه نویسی C (21814) - فصل سوم

مثال:

**printf("one\ntwo");    prints**

**one**

**two**

**printf("one\ttwo"); prints**

**one    two**

**printf("one two three"); prints**

**one two    three**

## زبان برنامه نویسی C (21814) - فصل سوم

مثال:

```
/* This Program Reads and echoes a single number .*/  
#include <stdio.h>  
void main (void){  
    int number;      //to store the Number  
  
    puts( "Please type a single number and hit enter");  
    scanf( "%i" ,&number);  
    printf("You typed  %i.\n",number);  
}
```

## زبان برنامه نویسی C (21814) - فصل سوم

### Print Specifiers and Parameters

- با استفاده از دستور `printf()` می توان در یک زمان چندین خروجی داشت:

`printf("%s\n%s", "one", "two");`

parameter one      parameter two      parameter three

which prints:

**one**

**two**



## زبان برنامه نویسی C (21814) - فصل سوم

### تمرین:

برنامه ای بنویسید که شماره دانشجویی شما را گرفته و سپس به روی خروجی چاپ کند.

## زبان برنامه نویسی C (21814) - فصل سوم

### The Big Picture

- The source of a C-Program is a text file – called *source code*.
- The compiler reads the source and reports errors.
- Execution starts at beginning of *main()* function.
- The program executes line by line.

## زبان برنامه نویسی C (21814) - فصل سوم

### Some Details

- C is case sensitive (printf not same as Printf).
- ***White Space*** in the source code is ignored by the compiler (program).
  - Spaces, tabs, newline (Enter key).
- Things have names.
  - include, main, printf, return.
- Punctuation, other symbols have specific meaning.
  - # ; , < > { } %

## زبان برنامه نویسی C (21814) - فصل سوم

### تمرین شماره یک

- نحوه نام گذاری فایل
- Comment نویسی و Indentation فراموش نشود.

## زبان برنامه نویسی C (21814) - فصل سوم

### محاسبات ساده

- پس از آموزش نحوه ورود و خروج داده ها از حافظه کامپیوتر، نحوه انجام محاسبات ساده بیان می شود.
- اعداد: اعداد را می توان به چندین صورت در حافظه کامپیوتر ذخیره نمود.
- اعداد صحیح (integer Type) که برای ذخیره اعداد صحیح بکار می رود. (این اعداد می تواند مثبت یا منفی باشند).

**int** Numbers;

- اعداد اعشاری (double Type) که برای ذخیره اعداد اعشاری بکار می رود.

**double** Ratio;

## زبان برنامه نویسی C (21814) - فصل سوم

مثال:

```
double Ratio;  
printf ("Please enter the Ratio:");  
scanf( "%lg", &Ratio);  
printf( "You entered %g \n", Ratio);
```

## زبان برنامه نویسی C (21814) - فصل سوم

روشهای تخصیص یک مقدار به یک متغیر:

- دریافت مقدار از طریق User با دستور `scanf`
- اختصاص مقدار در داخل برنامه

**Ratio = 5;**

## زبان برنامه نویسی C (21814) - فصل سوم

### مقادیر ثابت: (Constants)

- **Constant** ها در داخل برنامه مقادیری هستند که یکبار در ابتدای برنامه تعریف شده و تغییر نمی کنند. روش تعریف آنها بشرح زیر است:

```
#define PI 3.1416
```

- در هنگام استفاده از دستور **define** توجه به نکات زیر ضروری است:

- No semicolon
- No = sign



## زبان برنامه نویسی C (21814) - فصل سوم

### فرمول های محاسباتی:

- فرمول های محاسباتی، که **expressions** نامیده می شوند، به راحتی نوشته می شوند:

$$\text{Area} = \text{PI} * r * r ;$$

## زبان برنامه نویسی C (21814) - فصل سوم

مثال:

محاسبه مجموع و میانگین سه عدد:

```
/* This is a code to calculate the summation as well as the average of three  
arbitrary numbers */
```

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

```
#define Multiplier 10
```

```
void main (void){
```

```
    double A,B,C;           // Define three variables to get the values
```

```
    double Sum, Average;
```

```
    printf(" Please enter three numbers\n");
```

```
    scanf(" %lg%lg%lg", &A,&B,&C);
```

```
    Sum= A+B+C;
```

```
    Average=Sum/3;
```

```
    printf("\n The Summation of the numbers is %g\n",Sum);
```

```
    printf("\n The Average of the numbers is %g\n",Average);
```

```
}
```

## زبان برنامه نویسی C (21814) - فصل سوم

**مثال:**

**محاسبه مساحت یک دایره**



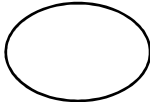
## زبان برنامه نویسی C (21814) - فصل سوم

### تمرین:

برنامه ای بنویسید که مربع و مکعب اعداد 1،2،3،4 را محاسبه و نمایش دهد.

## زبان برنامه نویسی C (21814) - فصل سوم

### نمودار جریان (Flow Chart)

	عملیات
	وجود خروجی های متفاوت در اثر مقایسه
	آغاز و خاتمه برنامه

## زبان برنامه نویسی C (21814) - فصل سوم

### حالت های شرطی:

- یکی از مزایای اصلی استفاده از کامپیوتر قابلیت آن در انجام واکنش های مختلف در حالت های متفاوت است.
- یکی از این روشها استفاده از دستورات `if` و `else if` است.

## زبان برنامه نویسی C (21814) - فصل سوم

### **:if Statement**

- در یک if Statement ساده پس از کلمه شرطی if، شرط در داخل پرانتز بیان می شود.
- پس از پرانتز در داخل { } دستوراتی که در صورت صحیح بودن شرط بایستی اجرا شوند، نوشته می شوند.
- در هنگام اجرای برنامه، در صورتیکه شرط برنامه صحیح باشد، دستورات داخل { } اجرا می شوند. در غیر اینصورت نادیده گرفته می شوند.

## زبان برنامه نویسی C (21814) - فصل سوم

**/\* This Program Reads and echoes a single number. \*/**

**#include <stdio.h>**

**void main (void){**

**int number; //to store the Number**

**printf( "Please type a single number: ");**

**scanf( "%i" ,&number);**

**if(number >100){**

**printf("\n\n You typed %i.\n",number);**

**printf("\n\n You typed a number greater than 100.\n");**

**}**

**printf("\n Have a good day.\n");**

**}**



## زبان برنامه نویسی C (21814) - فصل سوم

### **:if ... else Statement**

- اگر مایل باشیم در صورت درست نبودن شرط، دستورات دیگری اجرا شوند، از عبارت فوق استفاده می کنیم.

## زبان برنامه نویسی C (21814) - فصل سوم

**/\* This Program Reads and echoes a single number. \*/**

**#include <stdio.h>**

**void main (void){**

**int** number;      //to store the Number

**printf( "Please type a single number: ");**

**scanf( "%i" ,&number);**

**if(number >100){**

**printf("\n\n You typed %i.\n",number);**

**printf("\n\n You typed a number greater than 100.\n");**

**}**

**else{**

**printf("\n\n You typed %i.\n",number);**

**printf("\n\n You typed a number less than 100.\n");**

**}**

**printf("\n Have a good day.\n");**

**}**

## زبان برنامه نویسی C (21814) - فصل سوم

### if ... else Statement (ادامه):

- عبارت if ... else می تواند بصورت مکرر در صورت وجود شرط های مختلف به کار رود.
- اگر فقط بخواهیم یک خط را در صورت صحیح بودن گزاره if اجرا کنیم، می توانیم براکت ها { } را حذف کنیم.

## زبان برنامه نویسی C (21814) - فصل سوم

```
#include <stdio.h>
void main (void){
    int number;          //to store the Number

    printf( "Please type a single number: ");
    scanf( "%i" ,&number);

    if(number >100){
        printf("\n\n  You typed  %i.\n",number);
        printf("\n\n  You typed a number greater than 100.\n");
    }
    else if (number<100){
        printf("\n\n  You typed  %i.\n",number);
        printf("\n\n  You typed a number less than 100.\n");
    }
    else{
        printf("\n\n  You typed  %i.\n",number);
        printf("\n\n  You typed a number equal to 100.\n");
    }
    printf("\n Have a good day.\n");
}
```

## زبان برنامه نویسی C (21814) - فصل سوم

### مثال:

برنامه ای بنویسید که درجه حرارت هوا را بصورت فارنهایت از کاربر گرفته و پس از محاسبات لازم اعلام نماید که درجه حرارت به مقیاس سانتی گراد بزرگتر، مساوی، و یا کوچکتر از صفر می باشد.

## زبان برنامه نویسی C (21814) - فصل سوم

### حلقه ها و تکرار ها:

- یکی از مزایای برنامه نویسی استفاده از لوپ (حلقه های تکرار) می باشد که اجازه می دهد که گروهی از دستورات را چندین مرتبه تکرار کنیم. این حلقه ها برنامه نویسی را ساده می کنند.

- زبان برنامه نویسی C شامل سه نوع حلقه می باشد:

- while
- do
- for

## زبان برنامه نویسی C (21814) - فصل سوم

**/\* This Program fills a screen with number lines. \*/**

**#include <stdio.h>**

**void main (void){**

**int number; //to store the Number**

**printf( "Please type a single number: ");**

**scanf( "%i" ,&number);**

**while(number>0){**

**printf(" %i \t",number);**

**number=number-1;**

**}**

**printf("\n The code is terminated\n");**

**}**

## زبان برنامه نویسی C (21814) - فصل سوم

### دستور **do ... while**

فرم عمومی این دستور به شرح زیر است:

```
do{  
    statements  
}while (expression)
```

- حلقه **do** اجرای **statement** ها را تا هنگامی که شرط **expression** صحیح باشد تکرار می کند.

- مثال: طراحی یک تست چهار جوابی



## زبان برنامه نویسی C (21814) - فصل سوم

**/\* This Program asks a question \*/**

**# include <stdio.h>**

**# include <conio.h>**

**void main (void){**

**char Answer ;**

**printf ( "Please Answer this question:\n\n " );**

**printf ( "What is the capital of Iran?\n\n " );**

**printf ( " A: Isfahan\n B: Tabriz\n C: Mashhad\n D: Tehran\n\n");**

**do{**

**Answer = getche();**

**}while( Answer!='A' && Answer!='B' && Answer!='C' && Answer!='D' );**

**printf("\n You have answered the question.\n\n");**

**if(Answer=='D'){**

**printf(" \*\*\*\* Your answer is correct. Good job. \*\*\*\*\n\n");**

**}**

**else{**

**printf(" \*\*\*\* Your answer is wrong!!! Study more \*\*\*\*\n\n");**

**}**

**}**

## زبان برنامه نویسی C (21814) - فصل سوم

### حلقه for:

حلقه for تقریباً همان وظایف حلقه های قبلی را بصورت متفاوتی انجام می دهد.

**مثال:** محاسبه مربع و مکعب اعداد 1 الی 4

## زبان برنامه نویسی C (21814) - فصل سوم

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

    int number, Square, Cube;    //to store the Number
    int i;
    printf( "Please type a single number: \n\n");
    scanf( "%i" ,&number);

    for (i = 1; i <= number; i++){
        Square = i*i;
        Cube = Square *i;
        printf( " The Square and Cube of %i\t are %i\t and %i,
        respectively.\n" ,i, Square, Cube);

    }
}
```

## زبان برنامه نویسی C (21814) - فصل سوم

### تمرین:

برنامه ای بنویسید که در آن اعداد زوج در فاصله 1 الی 1000 در خروجی کامپیوتر چاپ شود.

## زبان برنامه نویسی C (21814) - فصل سوم

### تمرین:

برنامه ای بنویسید که در آن از میان یکسری اعداد داده شده توسط کاربر (تعداد آنها توسط کاربر تعیین می شود)، مقادیر حداکثر و حداقل آنها را محاسبه و چاپ نماید.

## زبان برنامه نویسی C (21814) - فصل سوم

### تمرین:

برنامه ای بنویسید که دترمینان یک ماتریس مربع  $3 \times 3$  را محاسبه نماید.



بنام خدا

زبان برنامه نویسی C (21814)

Lecture 3

Chapter 4

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814) نحوه نوشتن یک برنامه

```
/*  
* Name: Nasser Salmasi  
* Date: 10/3/06  
* Assignment: Term Project1  
* Description: Johnson's Algorithm.  
*/  
  
#include <stdio.h>  
  
int main(void) {  
  
    /* print the message */  
    printf("Go Beavers!");  
    system("PAUSE");  
    return 0;  
}
```

**File Header Comment**

**Blank Lines (readability)**

**Inline Comment**

**Indenting – Use Tab Key!**



## زبان برنامه نویسی C (21814) نحوه نوشتن یک برنامه

- Print the squares and cubes of integers from 1 through 3.
- Declare variables needed.
- Calculate the numbers.
- Print the results.

## زبان برنامه نویسی C (21814) نحوه نوشتن یک برنامه

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

```
int main(void) {
```

```
    int n, square, cube;    /* n is input, square & cube are output */
```

```
    printf("Table of squares and cubes\n");
```

```
    n = 1;
```

```
    square = n * n; cube = n * square; /* first line of output */
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    n = 2;
```

 **note**

```
    square = n * n; cube = n * square; /* second line of output */
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    n = 3;
```

```
    square = n * n; cube = n * square; /* third line of output */
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    return 0;
```

```
}
```

```
/*  
 * Name: Nasser Salmasi  
 * Instructor: Nasser Salmasi  
 * Date: Oct 6, 2007  
 * Assignment: SquaresCubes  
 * Description: Make a table of the integers from 1 – 3 and their squares and cubes.  
 */
```

header comment

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

readability  
(blank lines)

inline comments

```
int main(void) {
```

```
    int n, square, cube;
```

// n is input, square & cube are output

```
    printf("Table of squares and cubes\n");
```

```
    n = 1;
```

```
    square = n * n; cube = n * square; // first line of output
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    n = 2;
```

```
    square = n * n; cube = n * square; // second line of output
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    n = 3;
```

```
    square = n * n; cube = n * square; // third line of output
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    return 0;
```

indentation -- Use Tab Key!

```
}
```

## زبان برنامه نویسی C (21814) نحوه نوشتن یک برنامه

همان برنامه بصورت نا مرتب

```
#include <stdio.h>
int main(void) { int n, square, cube;
printf("Table of squares and cubes\n");
n = 1; square = n * n; cube = n * square;
printf("%d %d %d\n", n, square, cube); n = 2;
square = n * n; cube = n * square;
printf("%d %d %d\n", n, square, cube); n = 3;
square = n * n; cube = n * square;
printf("%d %d %d\n", n, square, cube);
return 0; }
```

## زبان برنامه نویسی C (21814)

### فصل چهارم: Objects, Types, and Expressions

در این فصل نحوه نام گذاری اجزای یک برنامه بررسی می شود.

## زبان برنامه نویسی C (21814)- فصل چهارم

### متغیر ها:

- یک متغیر مکانی است برای ذخیره سازی داده ها که در هر زمان قادر است یک داده را ذخیره نماید.
- بر اساس نوع داده ای که در متغیر ذخیره می شود، نوع متغیر (Type or data type) تعریف می گردد.
- هر متغیری قبل از استفاده بایستی با declaration تعریف شود.

## زبان برنامه نویسی C (21814)- فصل چهارم

### متغیر ها (ادامه):

- در تعریف یک متغیر نام، نوع متغیر، و مکان آن در حافظه تعریف می شود.
- هنگام تعریف متغیر ها بسیار مهم است که مقدار اولیه قبل از استفاده از آن تخصیص داده شود.
- نام متغیر بایستی متناسب با نوع مقادیری باشد که در آن ذخیره می شود.

## زبان برنامه نویسی C (21814) - فصل چهارم

### ثابت ها (Constants):

- ثابت ها اجزایی از برنامه هستند که مقدارشان در طول برنامه تغییر نمی کند.
- در زبان برنامه نویسی C ثابت ها به دو صورت تعریف می شوند:
  - استفاده از دستور `#define` (می توان آن را بصورت دستی تنظیم کرد)
  - استفاده از دستور `const` (مطالعه بر عهده دانشجو).



## زبان برنامه نویسی C (21814)- فصل چهارم

### قوانین نام گذاری اجزای برنامه:

- نام ها بایستی با حروف یا کاراکتر \_ آغاز شوند.
- نام ها می توانند تنها شامل حروف، اعداد، و یا \_ باشند.
- نمی توان کلماتی را که در زبان برنامه نویسی C تعریف شده اند را به عنوان کلمه جدید تعریف کرد (double, while,...).

## زبان برنامه نویسی C (21814)- فصل چهارم

### قوانین نام گذاری اجزای برنامه (ادامه):

- زبان برنامه نویسی C نسبت به حروف کوچک و بزرگ حساس است (Germ != germ).
- در بعضی از کامپایلر های قدیمی نام متغیر نمی تواند بیش از 8 کاراکتر باشد.
- راهنمای نام گذاری متغیر ها در صفحه 100 کتاب مطالعه شود.

## زبان برنامه نویسی C (21814) - فصل چهارم

### اپراتورهای محاسباتی:

- زبان برنامه نویسی C قادر به درک مفهوم چهار عمل اصلی (+, -, \*, /) می باشد.
- اولویت استفاده از این عوامل اصلی به اینصورت است که \* و / دارای اولویت بالاتری می باشند.
- در صورتیکه بخواهیم اولویت ها را تغییر دهیم، بایستی از پرانتز استفاده کنیم.

## زبان برنامه نویسی C (21814) - فصل چهارم

### اپراتورهای محاسباتی (ادامه):

- زبان C دارای اپراتورهای متفاوتی است که هر یک کارهای متفاوتی را انجام می دهند.
- اولویت ها و رابطه میان این اپراتورها در ضمیمه B (Appendix B) ارائه شده است.

## زبان برنامه نویسی C (21814) - فصل چهارم

اپراتورهای محاسباتی (ادامه):

**مثال:**

$$3 / Z * 10 \% 3$$

$$(3 / Z) * 10 \% 3$$

$$((3 / Z) * 10) \% 3$$

$$(((3 / Z) * 10) \% 3)$$

## زبان برنامه نویسی C (21814) - فصل چهارم

اپراتورهای محاسباتی (ادامه):

مثال:

$$X = 3 * Y + 2 - 4 * Y / 14$$

$$X = 3 * (Y + 2) - 4 * (Y / 14)$$

## زبان برنامه نویسی C (21814) - فصل چهارم

اپراتورهای محاسباتی (ادامه):

**تمرین:** مقدار متغیر **X** را در روابط زیر محاسبه کنید:

$$Y = 4 \quad Z = 5 \quad K = 17$$

$$X = 3 * Y + 8 / Y - K / Z * 5$$

$$X = Y / Z * 1000 * K + 500 * Y / Z$$

## زبان برنامه نویسی C (21814) - فصل چهارم

اپراتور های محاسباتی (ادامه):

Symbol	Example	Meaning
=	$X = 6$	ذخیره مقدار 6 در X
+=	$X += 5.5$	$X = X + 5.5$
-=	$X -= 2$	$X = X - 2$
*=	$X *= 8$	$X = X * 8$
/=	$X /= 2$	$X = X / 2$



## زبان برنامه نویسی C (21814) - فصل چهارم

اپراتورهای محاسباتی (ادامه):

**تمرین:** در برنامه زیر مقادیر نهایی متغیرهای a,b,c چقدر است؟

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

```
int a = 12, b = 16, c = 100 , i;
```

```
void main (void){
```

```
    for(i = 1; i <= 5; i++){
```

```
        a += 1;
```

```
        b -= 2;
```

```
        c += 3;
```

```
    }
```

```
    printf(" a= %i \n b= %i \n c= %i \n\n ", a,b,c);
```

```
}
```

## زبان برنامه نویسی C (21814) - فصل چهارم

### اپراتور های افزایشی و کاهشی :

Fixity	Symbol	Example	Meaning
Postfix	++	<b>i++</b>	مقدار <i>i</i> استفاده شده سپس یکی اضافه می شود.
	--	<b>i--</b>	مقدار <i>i</i> استفاده شده سپس یکی کم می شود.
Prefix	++	<b>++i</b>	ابتدا به مقدار <i>i</i> یکی اضافه شده سپس استفاده می شود.
	--	<b>--i</b>	ابتدا از مقدار <i>i</i> یکی کم شده سپس استفاده می شود.

## زبان برنامه نویسی C (21814) - فصل چهارم

**مثال:**

در حین برنامه نویسی می توان به جای  $X=X+1$  از دستور زیر استفاده کرد:

**$X = X+1;$     ~     $++X;$**

## زبان برنامه نویسی C (21814) - فصل چهارم

اپراتورهای محاسباتی (ادامه):

**تمرین:** در برنامه زیر مقادیر نهایی متغیرهای a,b,c چقدر است؟

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

```
int a = 12, b = 16, c = 100,i;
```

```
void main (void){
```

```
    for(i=1; i<=5; i++){
```

```
        ++a;
```

```
        --c;
```

```
        ++b;
```

```
        a++;
```

```
    }
```

```
    printf(" a= %i \n b= %i \n c= %i \n\n ", a,b,c);
```

```
}
```

## زبان برنامه نویسی C (21814) - فصل چهارم

اپراتورهای محاسباتی (ادامه):

**تمرین:** در برنامه زیر مقادیر نهایی متغیر های a,b,c چقدر است؟

```
int a = 5, b = 7, c = 11;
```

```
printf( " a = %i   b = %i   c = %i", ++a, b--, c++);
```

## زبان برنامه نویسی C (21814) - فصل چهارم

اپراتورهای محاسباتی (ادامه):

**تمرین:** مقدار **d** در رابطه های زیر را بیابید:

```
int a = -2, b = 3, c = 4, d = 5;
```

```
d *= a-- + c*(++d) - b*(--b+2);
```

```
int a = 5, b = 6, c = 4, d = 5;
```

```
d *= 2* a-- + c*(++d) - b*(--b+2);
```

## زبان برنامه نویسی C (21814)- فصل چهارم

### اپراتور های مقایسه ای (Relational Operators):

- اپراتورهای مقایسه ای معمولاً برای مقایسه دو مقدار و ارائه پاسخ صحیح یا غلط به کار می روند.
- یکی از کاربرد های این اپراتور ها، دستور if می باشد.

## زبان برنامه نویسی C (21814) - فصل چهارم

اپراتور های مقایسه ای (ادامه) :

Usage	Meaning
$X < Y$	
$X \leq Y$	
$X > Y$	
$X \geq Y$	
$X \neq Y$	
$X == Y$	*****



## زبان برنامه نویسی C (21814)- فصل چهارم

اپراتور های مقایسه ای (ادامه) :

- یکی از اشتباهات بسیار متداول در برنامه نویسی جایگذاری اشتباه = با == است.

## زبان برنامه نویسی C (21814) - فصل چهارم

### متغیرهای منطقی (Logical Operators):

Operands		Results		
X	Y	!X	X && Y	X    Y
0	0	T	0	0
0	T	T	0	T
T	0	0	0	T
T	T	0	T	T

## زبان برنامه نویسی C (21814) - فصل چهارم

### تمرین:

در برنامه زیر مقدار متغیر i در دستور printf چقدر است؟

```
void main(void) {  
  
    int    n,i =0,k = 2;  
    for ( n = 0; n < 6; n++ , i++)  
        ;  
    printf( "\n %i \n\n", i );  
}
```

## زبان برنامه نویسی C (21814)- فصل چهارم

### تمرین:

برنامه ای بنویسید که مختصات يك نقطه در صفحه مختصات دکارتی را از کاربر دریافت و پیغامی چاپ کند که نقطه روی کدام محور مختصات و یا در کدام ربع صفحه مختصات قرار دارد.

## زبان برنامه نویسی C (21814)

### مطالبی در مورد پروژه اول:

#### گزارش نویسی:

- گزارش کامل شامل مقدمه، شرح مساله، ارایه روش حل و توضیح الگوریتم.
- استفاده صحیح از مراجع با ذکر ماخذ.
- دستور زبان و املاي صحیح و اشکالات تایپی

## زبان برنامه نویسی C (21814)

### مطالبی در مورد پروژه اول:

#### گزارش نویسی:

- برداشت شخصی، نه رونویسی
- گزارش از نظر فیزیکی منسجم باشد (حداقل منگنه شود!!).
- در هر گزارشی حتما نتیجه گیری لازم است.

## زبان برنامه نویسی C (21814)

مطالبی در مورد پروژه اول:

گزارش نویسی:

- گزارش نیاز به شماره صفحه دارد.

- فهرست

- صفحه اول گزارش شماره صفحه وارد نمی شود.

بِنامِ خدا



دانشگاه صنعتی شریف

زبان برنامه نویسی C (21814)

Lecture 4

Chapter 5

مدرس: ناصر سلماسی



## زبان برنامه نویسی C (21814) - فصل پنجم

### بکارگیری Function ها و Library ها

- در این فصل مهم ترین ابزار برنامه نویسی که برنامه را قابل کنترل می کند ( Function ها) بررسی می شود.
- در واقع می توان به کمک Function ها برنامه های بزرگ را به اجزای کوچکتر قابل کنترل تجزیه نمود.
- یک Function در واقع قسمتی از برنامه است که دارای نامی بوده و هنگامی که صدا می شود وظیفه مشخصی را انجام می دهد.

## زبان برنامه نویسی C (21814) - فصل پنجم

### Function ها (ادامه):

- **Function** هایی که تاکنون مورد بررسی قرار گرفته اند عبارتند از:

- **puts()**
- **printf()**
- **scanf()**
- **main()**

## زبان برنامه نویسی C (21814) - فصل پنجم

### Function ها (ادامه):

- نوشتن یک برنامه همانند ساختن یک کامپیوتر است.
- در یک برنامه مرتب، هدف یا وظیفه هر Function کاملاً مشخص است.
- هیچ تابعی نبایستی بصورت طولانی و پیچیده نوشته شود و بایستی به اجزای کوچکتر شکسته شود.

## زبان برنامه نویسی C (21814) - فصل پنجم

### Library ها:

- در زبان C، Function های مختلفی از منابع متفاوتی استفاده می شوند.
- بسیاری از این توابع بخشی از Library هایی هستند که بصورت استاندارد در زبان C تعریف شده اند.
- استفاده از این توابع به سرعت و قابلیت اعتماد برنامه کمک شایانی می کند.
- در زبان C، کاربر می تواند توابعی را که فعالیت های مورد نظر خود را انجام می دهند تعریف کند.

## زبان برنامه نویسی C (21814) - فصل پنجم

### Library های استاندارد:

- در زبان C در حدود 12 Standard library وجود دارد.
- معروف ترین و مهم ترین آنها همان `stdio.h` است که مورد بحث قرار گرفت.
- همانطور که قبلا بیان شد، در این `library`، روش های دریافت و ارسال داده ها تعریف شده است.

## زبان برنامه نویسی C (21814) - فصل پنجم

### Library های استاندارد:

- دومین Standard library مهم “math” می باشد که شامل تعدادی از توابع ریاضی می باشد.
- فهرست این توابع در جدول 5.1 کتاب در صفحه 140 موجود است.
- تعدادی از مهمترین این توابع عبارتند از:

## زبان برنامه نویسی C (21814) - فصل پنجم

### Library <math.h>

- **pow(base, exp)** example.  
value = pow((1+i), -n); // i.e. value =  $(1+i)^{-n}$
- **sqrt(value)** example.  
root = sqrt (x\*x + y\*y); // i.e.  $root = \sqrt{x^2 + y^2}$
- **fabs(value)** example;  
err = fabs((x-xe)/x); // i.e. err =  $|(x-xe)/x|$
- **floor(value)** example;  
ABS=floor(x);

## زبان برنامه نویسی C (21814) - فصل پنجم

**Library <math.h> (ادامه):**

**تمرین:**

برنامه ای بنویسید که ریشه های یک معادله درجه دو را محاسبه نماید.



## زبان برنامه نویسی C (21814) - فصل پنجم

مثال:

```
#include <stdio.h>
#include <math.h>
void main (void){
    int A,B,C; // Define three variables to get the values

    printf(" Please enter Value for A:\n");
    scanf(" %i", &A);
    printf(" Please enter Value for B:\n");
    scanf(" %i", &B);
    printf(" Please enter Value for C:\n");
    scanf(" %i", &C);

    printf("\n The 1st root is %f \n",(-B + sqrt(pow(B, 2)- 4*A*C) )/(2*A));
    printf("\n The 2nd root is %f \n",(-B - sqrt(pow(B, 2)- 4*A*C) )/(2*A));

}
```

## زبان برنامه نویسی C (21814) - فصل پنجم

- سومین **Standard library** مهم **"stdlib"** می باشد.
- مهمترین کاربرد آن برای دانشجویان مهندسی صنایع تولید اعداد تصادفی است.
- کاربرد فراوان در دروس مهندسی صنایع و تحقیقات مربوط به آن.

## زبان برنامه نویسی C (21814) - فصل پنجم

- **Standard library** های مهم دیگر عبارتند از:
- **time**
- **string**

## زبان برنامه نویسی C (21814) - فصل پنجم

- **Standard library** ها بسیاری از مشکلات عمومی برنامه نویسی را مرتفع می کنند.
- **Local libraries (مانند tools).**
- این توابع در دسترس نمی باشند.

## زبان برنامه نویسی C (21814) - فصل پنجم

- هر یک از Standard library ها یک header file برای خود را دارد که با .h خاتمه می یابد.
- header file شامل prototype تمام توابعی است که در library تعریف شده اند. تعدادی از آنها عبارتند از:

Standard I/O library:	<stdio.h>	Standard library	<stdlib.h>
Mathematical library:	<math.h>	Time library	<time.h>
Character handling:	<ctype.h>	String library	<string.h>
Local tools library:	“tools.h”		

## زبان برنامه نویسی C (21814) - فصل پنجم

### Function ها:

- هر تابعی بایستی Type مشخصی داشته باشد که به آن prototype می گویند که بایستی قبل از بکار گیری آن در برنامه تعریف شود.
- هنگامی که یک تابع در برنامه صدا زده می شود، کنترل برنامه به ابتدای متن تابع رفته متن آن را اجرا می کند.
- پس از اجرای تابع، مجددا کنترل برنامه به خط بعدی که تابع صدا زده شده است می رود.

## زبان برنامه نویسی C (21814) - فصل پنجم

### Function ها (ادامه):

- برای صدا زدن یک تابع نام تابع را به همراه یک جفت ( ) که داخل آن می تواند تهی یا شامل چندین argument باشد را می نویسند.
- در کتاب دستوری که تابع را صدا می زند را caller و تابع صدا زده شده را subprogram و یا function می نامند.
- Argument های مورد نیاز توسط caller مشخص و به تابع فرستاده می شود.

## زبان برنامه نویسی C (21814) - فصل پنجم

### Function ها (ادامه):

- هنگام صدا زدن تابع، در caller مربوطه بایستی مقداری برای هر پارامتر تعریف شده در تابع ارائه شود. نمونه آن به شرح زیر است (صفحات 142 و 143 مطالعه شود):

<b>double</b>	<b>drop</b>	<b>(double</b>	<b>height);</b>
↑	↑	↑	↑
<b>Return</b>	<b>Function</b>	<b>argument</b>	<b>parameter</b>
<b>Type</b>	<b>name</b>	<b>type</b>	<b>name</b>



## زبان برنامه نویسی C (21814) - فصل پنجم

### توابع ایجاد شده توسط کاربر (User-Defined Functions):

دلایل استفاده از توابع در برنامه نویسی:

- آسان سازی درک برنامه نوشته شده برای سایرین
- امکان بکار گیری تمام برنامه یا قسمتی از آن در جاهای دیگر
- شکسته شدن یک برنامه به اجزای کوچکتر به منظور کنترل بهتر برنامه

## زبان برنامه نویسی C (21814) - فصل پنجم

یک تابع شامل قسمت های زیر می باشد:

- **Prototype (\*)**
- **Definition**
  - **Function header** (بایستی با \* هماهنگ باشد)
  - **Function body**

**مثال:** if....else برنامه

## زبان برنامه نویسی C (21814) - فصل پنجم

- روش دوم تعریف تابع:

**Double: Double function**

- برای مطالعه به بخش 5.4.2 کتاب مراجعه کنید.
- استفاده از هر یک از انواع تعریف توابع اختیاری است.

## زبان برنامه نویسی C (21814) - فصل پنجم

### تمرین:

برنامه ای بنویسید که بتواند هر جمله یک تصاعد هندسی یا حسابی را محاسبه کند. برای این منظور برنامه بایستی از کاربر اطلاعات زیر را دریافت کند:

- جمله اول و قدر نسبت
  - نوع تصاعد (کاربر می تواند هر دو را انتخاب کند)
  - جمله مورد علاقه در تصاعد (یا تصاعد ها) و حد مجموع (در صورت درخواست تصاعد هندسی) آنها را محاسبه نماید.
- سپس برنامه پاسخ ها را به روی خروجی ارسال کند. این برنامه را با ایجاد Function های متعدد ایجاد کنید.



دانشگاه صنعتی شریف

بِنام خدا

زبان برنامه نویسی C (21814)

## Lecture 5

### Chapter 10 and 17

### Introduction to Arrays

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814)

### فصل دهم- مقدمه ای بر آرایه ها (Arrays)

- نوع داده هایی که تا کنون مورد بررسی قرار گرفتند به حدی ساده بودند که توسط متغیر های ساده قابل بررسی و پردازش بودند.
- در دنیای واقعی مسایل پیچیده تری وجود دارند که برای حل آنها نیاز به تعریف متغیر های فراوان و پیچیده تری است (جدول مند لیف).
- آرایه ها به حل چنین مسایلی کمک می کنند.
- در این جلسه، نحوه تعریف، دسترسی، و کنترل اجزای یک آرایه بررسی می شود.

## زبان برنامه نویسی C (21814) - آرایه ها

- در بسیاری از برنامه ها، هر یک از داده ها که در متغیری ذخیره شده است فقط یک مرتبه پس از خوانده شدن پردازش می شود.
- در بعضی از برنامه ها، ابتدا تمامی داده ها خوانده شده، محاسباتی به روی آنها انجام شده، و پردازش مجددی به روی آنها انجام می شود.
- در چنین حالت هایی، داده ها بایستی برای کاربرد مجدد در طول زمان میان خواندن و پردازش مجدد حفظ شوند.

## زبان برنامه نویسی C (21814) - آرایه ها

- یک آرایه شامل یک سری پیاپی از متغیر هاست که دارای نام یکسانی هستند.  $X_1, X_2, X_3, \dots, X_n$
- هر یک از این متغیر ها Array slots نامیده می شوند.
- مقداری که به هر متغیر اختصاص می یابد Array element نام دارد.
- تعداد فضایی که در یک آرایه یک بعدی ایجاد می شود طول آرایه نام دارد.



## زبان برنامه نویسی C (21814) - آرایه ها

- می توان آرایه های دو بعدی نیز تعریف نمود (فصل 17)
- نوع متغیر هایی که در یک آرایه تعریف می شوند بایستی یکسان باشد.
- تمامی متغیر های یک آرایه یک نام می گیرند. می توان در صورت نیاز به کل آرایه یا به یکی از اجزای آن رجوع کرد. ( $X[2]$ )
- در زبان برنامه نویسی C تمامی آرایه ها با اندیس صفر آغاز می شوند زیرا راحت تر و موثر تر است.

## زبان برنامه نویسی C (21814) - آرایه ها

### نحوه تعریف آرایه ها:

يك آرایه بسیار شبیه يك متغیر معمولي تعریف می شود.

```
int X[7];
```

?	?	?	?	?	?	?
X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]

## زبان برنامه نویسی C (21814) - آرایه ها

نحوه مقدار دهی به متغیر های آرایه:

```
int X[7] = {4,5,6,1,2,7,0};
```

4	5	6	1	2	7	0
X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]

## زبان برنامه نویسی C (21814) - آرایه ها

نحوه مقدار دهی به متغیر های آرایه:

```
int X[7] = {4,5};
```

4	5	0	0	0	0	0
X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]

## زبان برنامه نویسی C (21814) - آرایه ها

نحوه مقدار دهی به متغیر های آرایه:

- این روش با error رو به رو می شود (برخلاف مطلب کتاب).

```
int X[7] = {};
```

0	0	0	0	0	0	0
X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]

زبان برنامه نویسی C (21814) - آرایه ها

روش دیگری برای مقدار دادن به طول يك آرایه:

```
#define j 6  
int My_Array[j];
```

زبان برنامه نویسی C (21814) - آرایه ها

روش دیگری برای مقدار دادن به طول يك آرایه:

```
int odds[ ] = {3,5,7,9,11};
```

## زبان برنامه نویسی C (21814) - آرایه ها

### توجه:

در صورتیکه در حین اجرای برنامه از طول يك آرایه عدول کنیم، با پیغام error مواجه نمی شویم. ولی برنامه بصورت GIGO کار می کند.

```
int odds[ ] = {3,5,7,9,11};  
printf ("%i  %i ",odds[4],odds[7]);
```



## زبان برنامه نویسی C (21814) - آرایه ها

### نحوه بکار گیری آرایه ها:

- معمولا آرایه ها در حلقه ها کاربرد دارند.
- مثال: گرفتن نمرات دانشجویان يك کلاس و محاسبه میانگین و آرایه گزارش.

## زبان برنامه نویسی C (21814) - آرایه ها

آرایه های دو بعدی:

```
int My_Array[10][10];
```

```
Int My_Array[4][4] = {{3,2,1,0},  
                      {6,2,3,1},  
                      {6,5,7,9},  
                      {0,3,1,2}  
                      };
```

## زبان برنامه نویسی C (21814) - آرایه ها

آرایه های دو بعدی:

```
int a[2][3]={0,2,4,6,8,10};
```

## زبان برنامه نویسی C (21814) - آرایه ها

تمرین:

```
int myArray[ ] = {0, 2, 4, 6, 8, 10};
```

```
myArray[myArray[1]] = ?
```

## زبان برنامه نویسی C (21814) - آرایه ها

**تمرین:** برنامه زیر را در نظر بگیرید:

```
int X[3], Array1[10] = {0,3,5,6,5,2,8};  
X[0] = Array[5] + Array[9]*2;
```

کدامیک از گزینه های زیر صحیح است؟

- a)  $X[0] = 2$
- b)  $X[0] = 18$
- c)  $X[0] = 5$
- d)  $X[0] = 17$
- e) هیچکدام

## زبان برنامه نویسی C (21814) - آرایه ها

**تمرین:** برنامه زیر را در نظر بگیرید:

```
int X[3], Array1[10] = {0,3,5,6,5,2,8};  
X[1] = ++Array[2] + Array[0]*2;
```

کدامیک از گزینه های زیر صحیح است؟

- a)  $X[1] = 6$       b)  $X[1] = 12$   
c)  $X[1] = 5$       d)  $X[1] = 17$       e) هیچکدام

## زبان برنامه نویسی C (21814) - آرایه ها

**تمرین:** برنامه زیر را در نظر بگیرید:

```
int X[3],Array1[10] = {0,3,5,6,5,2,8};  
X[2] = Array[4]*(Array[10] + Array[9]*2);
```

کدامیک از گزینه های زیر صحیح است؟

- |                |                |            |
|----------------|----------------|------------|
| a) $X[2] = 0$  | b) $X[2] = 5$  |            |
| c) $X[2] = 14$ | d) $X[2] = 17$ | e) هیچکدام |

## زبان برنامه نویسی C (21814) - آرایه ها

کدامیک از گزینه های زیر در مورد مقدار دهی ابتدایی به آرایه **X** صحیح است؟

- a) `int X[2][2] = {};`
- b) `int X[2][2] = 5;`
- c) `int X[2][2]={2,3,4,0};`
- d) `int X[2] [2]= 0;`
- e) هیچکدام



## زبان برنامه نویسی C (21814) - آرایه ها

### تمرین:

آرایه دو بعدی زیر را در نظر بگیرید. مقدار متغیر  $x$  کدام است؟

```
int a[2][3]={0,1,4,6,8,0},x;
```

```
x = a[1][1]* a[a[0][1]][a[0][2]-2] + a[0][2];
```

الف-4      ب-5      ج-11      د-12

## زبان برنامه نویسی C (21814) - آرایه ها

### تمرین:

برنامه ای بنویسید که شماره دانشجویی هر دانشجو را ، به علاوه نمرات میان ترم و پایان ترم را بگیرد.

سپس میانگین نمرات میان ترم و پایان ترم را محاسبه و اختلاف هر یک از نمرات دانشجو را با میانگین اعلام نماید. همچنین واریانس میان ترم و پایان ترم را محاسبه کنید.



بِنامِ خدا

## زبان برنامه نویسی C (21814)

Lecture 6

Chapter 6

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814) - فصل ششم

### نکات ناگفته فصل ششم:

- همانطور که مشاهده شد حلقه های `for` و `while` حلقه هایی هستند که شرط خاتمه برنامه در ابتدا بررسی می شود.
- در حلقه `do... while` ابتدا متن حلقه اجرا شده، سپس شرط خاتمه بررسی می شود.
- در بعضی از برنامه ها لازم است که این شرط در داخل حلقه بررسی شود.

## زبان برنامه نویسی C (21814) - فصل ششم

### نکات ناگفته فصل ششم (ادامه):

در زبان برنامه نویسی C به سه صورت می توان کنترل برنامه را به مکانی دیگر از برنامه انتقال داد. این روشها عبارتند از:

- دستور break
- دستور continue
- دستور goto که توصیه می شود استفاده نشود.

- هیچکدام از این دستورات برای برنامه نویسی لازم نبوده و یک برنامه می تواند بدون این دستور ها نیز نوشته شود. ولی کاربرد آنها می تواند به ساده تر، کوتاه تر شده، و موثر تر شدن برنامه کمک کند.

## زبان برنامه نویسی C (21814) - فصل ششم

### دستور break:

- وظیفه دستور break قطع روال عادی اجرای برنامه در داخل یک حلقه و انتقال آن به پایان حلقه است.

- یکی از کاربردهای متداول آن در داخل یک حلقه do... while است که دستور break به عنوان گزینه ای در صورت صحیح بودن if statement به کار می رود.

```
do{  
....  
    if( ){  
        break;  
    }  
}
```

next statement // control comes here after the break

## زبان برنامه نویسی C (21814)

مثالی دیگر از دستور **break**:

```
for( initialization; condition; update){  
    statements  
    if (condition){  
        break;  
    }  
    more statements  
}
```

## زبان برنامه نویسی C (21814)

### دستور break (ادامه):

- استفاده از دستور break موجب کوتاه تر شدن طول برنامه، همچنین سریع تر شدن آن با کنار گذاشتن قسمت هایی که در صورت صحیح بودن if statement نیازی به آنها نیست می شود.

### • دو کاربرد مهم این دستور عبارتند از:

- هنگامی که هدف برنامه انجام شده است (حالت فوق).
- حالتی که مشکلی (ERROR) در برنامه بوجود آمده.



## زبان برنامه نویسی C (21814)

مثال:

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

    int R1,R2,k=0;
    do{
        printf("Please enter the result of the first dice: ");
        scanf("%i",&R1);
        printf("Please enter the result of the second dice: ");
        scanf("%i",&R2);
        if((R1>= 1 && R1<= 6) && (R2 >= 1 && R2 <= 6)){
            printf(" The summation is %i\n\n",R1+R2);
            break;
        }
        else{
            printf("\n\n Impossible Result. Try later!! \n\n");
        }
    } while(k<2);
}
```

## زبان برنامه نویسی C (21814)

### تمرین:

برنامه ای بنویسید که 10 عدد صحیح را از کاربر بگیرد. اگر در حین کار کاربر عدد صفر را به عنوان یکی از اعداد وارد کند، دیگر عدد دیگری وارد نشود. سپس اعداد وارد شده، بزرگترین عدد، کوچکترین، و میانگین آنها را چاپ نماید.

## زبان برنامه نویسی C (21814)

### دستور **continue**:

- وظیفه دستور **continue** قطع روال عادی اجرای برنامه در داخل یک حلقه و انتقال آن به آغاز حلقه می باشد.
- اگر این دستور داخل یک حلقه **for** باشد، پردازش برنامه با اضافه کردن مقدار شمارنده ادامه می یابد.
- اگر این دستور داخل یک حلقه **do ... while** باشد، پردازش برنامه با **loop test** ادامه می یابد.

## زبان برنامه نویسی C (21814)

مثال:

```
#include <stdio.h>

void main(void) {

    for (int i=1; i<=50; i++){
        if(i%2 != 0)
            continue;
        printf(" %i ", i);
    }
}
```

## زبان برنامه نویسی C (21814)

### **:Defective loops**

- حلقه بایستی از نظر تعریف شمارنده و شرط خاتمه حلقه صحیح تعریف شده باشند در غیر اینصورت ممکن است اجرای برنامه هیچگاه از حلقه خارج نشود.

## زبان برنامه نویسی C (21814)

### دستور Switch:

```
/* This Program Reads and echoes a single number. */  
# include <stdio.h>  
int number;      //to store the Number  
void main (void){  
    printf ( "Type an int number between 12 to 15:\n\n ");  
    scanf ( "%i" ,&number);  
    switch (number){  
        case 12: printf (" You entered 12.\n\n");   break;  
        case 13: printf (" You entered 13.\n\n");   break;  
        case 14: printf (" You entered 14.\n\n");   break;  
        case 15: printf (" You entered 15.\n\n");   break;  
        default: printf (" You entered a wrong number.\n");  
    }  
}
```

## زبان برنامه نویسی C (21814)

**مثال:** خروجی برنامه زیر چه خواهد بود؟

```
char c = 'S';  
switch (c) {  
    case 'T': printf("alpha"); break;  
    case 'S': printf("beta"); break;  
    default: printf("gamma");  
}
```

a) alpha

b) beta

c) gamma

d) beta gamma

e) هیچکدام

## زبان برنامه نویسی C (21814)

**مثال:** خروجی برنامه زیر کدام است؟

```
char a = 'b';  
switch(a) {  
    case 'a': printf("one"); break;  
    case 'b': printf("two"); break;  
    case 'c': printf("three"); break;  
    default: printf("nil");  
}
```

الف – one      ب- two      ج- nil      د - هیچکدام





بنام خدا

زبان برنامه نویسی C (21814)

Lecture 7

Chapter 7

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814)

### مطالبی در مورد پروژه اول:

#### گزارش: 30%

– مقدمه 5%

– صحیح بودن متن الگوریتم 5%

– ایراد املائی 3%

– ایراد علمی 5%

– نتیجه گیری 5%

– فهرست 2%

– حل مساله نمونه 5%

## زبان برنامه نویسی C (21814)

مطالبی در مورد پروژه اول:

نمودار جریان: 15%

برنامه: 55%

– اجرای برنامه بدون warning و Error 20%

– صحت برنامه و الگوریتم 20%

– مرتب بودن برنامه 10%

– داشتن comment مناسب 5%

## زبان برنامه نویسی C (21814) - فصل هفتم

### مطالبی در مورد انواع متغیر های عددی:

متغیرهای عددی در زبان C به دو صورت تعریف می شوند:

- بصورت **integer**:

- بصورت **float** :

## زبان برنامه نویسی C (21814) - فصل هفتم

### متغیر های integer:

- این نوع متغیرها دارای انواع مختلفی بوده که همگی برای ذخیره سازی متغیرهای عدد صحیح به کار می روند.
- نحوه تعریف انواع متغیر های integer و طول هر متغیر در جدول 7.1 صفحه 229 کتاب موجود است.
- با بکار گیری int ذخیره سازی یک عدد تا 10 رقم امکان پذیر است.

## زبان برنامه نویسی C (21814) - فصل هفتم

### متغیر های float :

- این نوع متغیر ها دارای انواع مختلفی بوده که همگی برای ذخیره سازی متغیر های عدد واقعی (real) به کار می روند.
- برای ساده سازی پیشنهاد می شود همواره در صورت نیاز به استفاده از متغیر float از حالت double که دارای بیشترین ظرفیت است استفاده شود.
- مقایسه ظرفیت حالت های مختلف در جدول 7.4 (صفحه 232) ارائه شده است.

## زبان برنامه نویسی C (21814) - فصل هفتم

### مطالبی در مورد انواع متغیر های عددی:

- در این فصل انواع متغیر های صحیح و واقعی که در زبان C به کار می روند، به همراه فرمت I/O آنها و محدوده پذیرش اعداد برای هر یک از آنها مورد بررسی قرار می گیرند.
- همچنین نحوه انتقال داده ها از یک نوع متغیر به نوع دیگر طی محاسبات ریاضی، به علاوه توصیه هایی برای انتخاب نوع متغیر مناسب صورت می گیرد.

## زبان برنامه نویسی C (21814) - فصل هفتم

### متغیر های اعداد واقعی:

در فرمت علمی یک عدد واقعی بصورت  $1.235 \times 10^7$  نوشته می شود. این اعداد را می توان به سه صورت:

- float
- double
- long double

نمایش داد.



## زبان برنامه نویسی C (21814) - فصل هفتم

متغیر های اعداد واقعی:

يك عدد float را مي توان به دو صورت نمایش داد:

**725.78**

**$7.2578 \times 10^2$  (base-10 scientific notation)**

## زبان برنامه نویسی C (21814) - فصل هفتم

### قوانین نوشتن اعداد float:

- يك عدد را مي توان با هر تعداد عدد اعشاري نوشت. هر عددي شامل نقطه اعشاري به عنوان float شناخته مي شود.

1.0, 0.1, 125.3, 0.123589, 1.

- مي توان اعداد را بصورت نماد علمي نيز نوشت. در اين قسمت بخش نمايي شامل يك حرف (E يا e)، يك علامت (اختياري) و سپس يك عدد (صحيح حداكثر سه رقمي) است.

4.6668E+7

## زبان برنامه نویسی C (21814) - فصل هفتم

### نحوه نوشتن و خواندن اعداد در زبان C:

- هنگام خواندن (دریافت) یک عدد به دو نکته بایستی توجه شود:
  - نوع متغیری که عدد خوانده شده در آن ذخیره می شود.
  - نوعی که متغیر وارد می شود.
- هنگام نوشتن (ارسال) یک عدد به دو نکته بایستی توجه شود:
  - نوع متغیری که عدد خوانده شده در آن ذخیره شده.
  - نوعی که متغیر ارسال می شود.

## زبان برنامه نویسی C (21814) - فصل هفتم

### نحوه نوشتن و خواندن اعداد صحیح در زبان C:

Context	Conversion	Meaning and Use
scanf()	%d	Read a base-10 integer
	%i	Read a decimal or Hexadecimal integer
printf()	%d	Print an integer in base 10
	%i	Same as %d for output

- بخش 7.3 کتاب مطالعه شود.

## زبان برنامه نویسی C (21814) - فصل هفتم

مثال:

```
#include<stdio.h>
void main (void){
    int A=12;
    printf( " A i based = %i\n", A );
    printf( " A d based = %d\n", A );

    printf("\nEnter Value for A: ");
    scanf("%i",&A);
    printf( " A i based = %i\n", A );
    printf( " A d based = %d\n", A );

    printf("\nEnter Value for A: ");
    scanf("%d",&A);
    printf( " A i based = %i\n", A );
    printf( " A d based = %d\n", A );
}
```

## زبان برنامه نویسی C (21814) - فصل هفتم

### نحوه نوشتن و خواندن اعداد واقعی در زبان C:

Context	Conversion	Meaning and Use
scanf()	%g, %f, or %e	Read a number and store in a <b>float</b> variable
	%lg, %lf, or %le	Read a number and store in a <b>double</b> variable
	%Lg, %Lf, %Le	Read a number and store in a <b>long double</b> variable
printf()	%f	Print a <b>float</b> or a <b>double</b> in decimal format
	%e	Print a <b>float</b> or a <b>double</b> in exponential format
	%g	Print a <b>float</b> or a <b>double</b> in general format

## زبان برنامه نویسی C (21814) - فصل هفتم

### مثال:

```
#include<stdio.h>
void main (void){
    double    A=12.3334546475;
    printf( " A f based = %f\n", A );
    printf( " A g based = %g\n", A );
    printf( " An e based = %e\n", A );
    printf("\nEnter Value for A f format:  ");
    scanf("%f",&A);
    printf( "\t A f based = %f\n", A );
    printf( "\t A g based = %g\n", A );
    printf( "\t An e based = %e\n", A );
    printf("\nEnter Value for A g format:  ");
    scanf("%g",&A);
    printf( "\t A f based = %f\n", A );
    printf( "\t A g based = %g\n", A );
    printf( "\t An e based = %e\n", A );
    printf("\nEnter Value for A e format:  ");
    scanf("%e",&A);
    printf( "\t A f based = %f\n", A );
    printf( "\t A g based = %g\n", A );
    printf( "\t An e based = %e\n", A );
}
```

## زبان برنامه نویسی C (21814) - فصل هفتم

### مثال:

Input at keyboard	%g	32.1786594	2.3	12345678
Internal bit Value		32.17865753173828125	2.2999999952316...	12345678
Output using	%f	32.178658	2.300000	12345678.00000
	%e	3.217866e+01	2.300000e+00	1.2345678e+07
	%g	32.1787	2.3	1.234567e+07
	%.3f	32.179	2.300	12345678.000
	%.3e	3.218e+01	2.300e+00	1.235e+07
	%.3g	32.2	2.3	1.23e+07
	%10.3f	32.179	2.300	12345678.000
	%-10.3f	32.179	2.300	12345678.000



## زبان برنامه نویسی C (21814) - فصل هفتم

مثال عملی:

## زبان برنامه نویسی C (21814) - فصل هفتم

مطالبی در مورد عملیات محاسباتی به روی متغیر ها:

**تقسیم:**

هنگام تقسیم دو عدد گاهی علاقمند به دانستن خارج قسمت هستیم،  
گاهی علاقمند به دانستن باقی مانده.

• برای محاسبه خارج قسمت دو متغیر از رابطه  $X/Y$  استفاده می کنیم.

## زبان برنامه نویسی C (21814) - فصل هفتم

مطالبی در مورد عملیات محاسباتی به روی متغیر ها (ادامه):  
مثال:

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

    int A=12, B=7,C;
    double D,E;

    C=A/B;
    D=A/B;
    E= (1.0)*A/B;

    printf(" %i   %i   %i\n\n", C,D,E);
    printf(" %f   %.5f   %f\n\n", C,D,E);
    printf(" %g   %g   %g\n\n", C,D,E);

}
```

## زبان برنامه نویسی C (21814) - فصل هفتم

مطالبی در مورد عملیات محاسباتی به روی متغیرها:

**تقسیم (ادامه):**

- برای محاسبه باقی مانده تقسیم دو عدد از رابطه  $X \% Y$  که همواره عددی بین 0 و  $Y-1$  است استفاده می کنیم.

## زبان برنامه نویسی C (21814) - فصل هفتم

مطالبی در مورد عملیات محاسباتی به روی متغیر ها (ادامه):  
مثال:

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

    int A=12, B=7,C;
    double D,E;

    C=A%B;
    D=A%B;
    E= (1.0)*A*B;

    printf(" %i   %i   %i\n\n", C,D,E);
    printf(" %f   %.5f   %f\n\n", C,D,E);
    printf(" %g   %g   %g\n\n", C,D,E);

}
```

## زبان برنامه نویسی C (21814) - فصل هفتم

مطالبی در مورد عملیات محاسباتی به روی متغیر ها:

**تقسیم (ادامه):**

- در مورد محاسبه باقی مانده تقسیم دو عدد از رابطه  $X \% Y$  توجه به نکات زیر ضروری است:

- همواره  $X \% Y = X - Y(X/Y)$ .
- اگر  $X$  مضربی از  $Y$  باشد،  $X \% Y = 0$ .
- اگر  $X < Y$  باشد،  $X \% Y = X$ .
- همواره مقدار  $X \% Y$  بین صفر و  $Y-1$  است.

## زبان برنامه نویسی C (21814) - فصل هفتم

مطالبی در مورد عملیات محاسباتی به روی متغیر ها(ادامه):

**تقسیم بر صفر:**

اگر در حین اجرای برنامه عددی بر صفر تقسیم شود، آنگاه برنامه با پیغام **ERROR** مواجه می گردد.

زبان برنامه نویسی C (21814) - فصل هفتم

مطالبی در مورد عملیات محاسباتی به روی متغیر ها:

**مطالعه بخش 7.4 توصیه می شود.**



## زبان برنامه نویسی C (21814) - فصل هفتم

### تمرین:

برنامه ای بنویسید که یک عدد صحیح چهار رقمی را بخواند و آن را بصورت زیر رمز گذاری نماید. هر رقم را با ”باقیمانده حاصل جمع آن رقم با 7 تقسیم بر 10“ جایگزین کند. سپس جای اولین رقم را با سومین رقم و جای دومین رقم را با چهارمین رقم عوض کند و عدد رمز شده را چاپ نماید.

## زبان برنامه نویسی C (21814) - فصل هفتم

### بخش 7.5:

### مشارکت دانشجویی

## زبان برنامه نویسی C (21814) - فصل هفتم

### مطالبی در مورد تولید اعداد تصادفی:

- اعداد تصادفی واقعی: ریختن تاس، پرتاب سکه ، انتخاب توپ از داخل يك کیسه شامل بي نهایت توپ.
- چنین عملی در برنامه نویسی و کارهای واقعی عملی نیست.
- استفاده از اعداد شبه تصادفی

## زبان برنامه نویسی C (21814) - فصل هفتم

### مطالبی در مورد تولید اعداد تصادفی:

- جدول اعداد شبه تصادفی در زبان های برنامه نویسی موجود است.
- هنگام استفاده از این جداول بایستی دقت شود.

## زبان برنامه نویسی C (21814) - فصل هفتم

### مثال:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
double R0,R1,R2;
void main(void){
    R0=rand();
    R1= rand();
    R2=R1/RAND_MAX;
    printf("\n \t %lg\t %lg\t %lg\t \n\n",R0,R1,R2);

    srand((int) time(NULL));
    printf("\n *** Second time ***\n");

    R0=rand();
    R1= rand();
    R2=R1/RAND_MAX;
    printf("\n \t %lg\t %lg\t %lg\t \n\n\n",R0,R1,R2);
}
```

## زبان برنامه نویسی C (21814) - فصل هفتم

### تمرین:

برای الگوریتم جانسون برنامه ای بنویسید که یک مساله شامل چندین کار (تعداد کارها تصادفی) را با زمان پردازش تصادفی به روی هر ماشین تولید نماید و نتایج را بصورت مرتب به روی خروجی کامپیوتر ارسال نماید.



بنام خدا

زبان برنامه نویسی C (21814)

Lecture 8

Chapters 8&9

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814) - فصل هشتم

### فصل هشتم:

#### نکاتی در مورد استفاده از اعداد

##### نکات اساسی این فصل:

- نحوه برخورد با تعدادی از مسایل محاسباتی نظیر نحوه مقایسه دو عدد
- نتایج خارج شدن مقدار اختصاص یافته به یک متغیر از محدوده قابل قبول



## زبان برنامه نویسی C (21814) - فصل هشتم

### مقایسه میان دو عدد:

- در حالت مقایسه دو عدد صحیح، مشکل چندانی وجود ندارد.
- مقایسه دو عدد واقعی در حالتی که نوع آنها یکسان نباشد، آسان نیست.

## زبان برنامه نویسی C (21814) - فصل هشتم

مثال:

```
#include <stdio.h>
float w = 4.4;
double x = 4.4;
void main (void){
    printf (" Is x==(double)w? %i    \n",(x == (double)w));
    printf (" Is (float)x == w? %i    \n",((float)x == w));
}
```

## زبان برنامه نویسی C (21814) - فصل نهم

- برنامه بایستی به اجزای کوچک (functions) شکسته شود.
- چیدمان عمومی یک برنامه بصورت زیر می باشد:
- **include commands for header files**
- **Constant definitions and type declarations**
- **Prototypes (function declarations) and one line functions**
- **main( ), which contains function calls**
- **Function definitions**

## زبان برنامه نویسی C (21814) - فصل نهم

- در مورد سایر توابع به جز `main ( )` اگر این توابع قبل از `main ( )` تعریف شوند، می توان `declaration` را برای آنها حذف نمود.
- معمولاً برنامه نویسان ترجیح می دهند که `main( )` به عنوان اولین تابع نوشته شود.

## زبان برنامه نویسی C (21814)

### مطالبی در مورد توابع (Functions):

تاکنون توابع مختلفی مورد بحث قرار گرفته اند:

- **int main(void)**
- **printf(arg1, arg2, ...)**
- **scanf(arg1, arg2, ...)**
- **sqrt(arg1)**
- **rand()**
- **time(NULL)**

## زبان برنامه نویسی C (21814)

### مطالبی در مورد توابع (Functions):

• فرآیند صدا زدن یک برنامه به شرح زیر است:

- انتقال اجرای برنامه به جای دیگری از برنامه
- انجام محاسبات یا برآوردن سرویس هایی دیگر برای برنامه
- بازگشت به مکانی که تابع صدا زده شده و ادامه کار.

## زبان برنامه نویسی C (21814)

روشهای استفاده از توابع:

- 1. `x = function1(arg1, arg2);`**  
takes arguments, returns a value.
- 2. `function2(arg1,arg2);`**  
takes arguments, returns nothing.
- 3. `function3();`**  
takes no arguments, returns nothing.

Last two examples sometimes called  
subroutine (when nothing returned)

## زبان برنامه نویسی C (21814)

Three parts to writing functions

1. **Declaration:** tell the compiler about the function (function prototype).
2. **Definition:** write the code the function executes.
3. **Call:** Call (use) the function to do some work.



## زبان برنامه نویسی C (21814)

### Declaration - Function Prototype

*type* function(*parameter\_list*);

for example:

double cube(double x); // x is optional

output value  
is type double  
(return value)

input value  
is type (double)

formal parameter name  
(optional)

## زبان برنامه نویسی C (21814)

### تعریف (Definition)

```
double cube(double val) {  
    return val*val*val;  
}
```

value to return

this function returns  
a value of type double

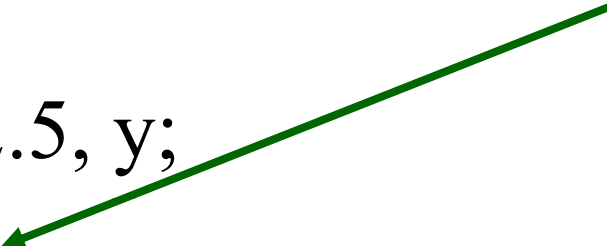
formal parameter --  
holds input value from  
call. **Implicitly declared  
and initialized variable!!**

## زبان برنامه نویسی C (21814)

### Call (Use)

Actual parameter -- the value  
passed to the function

```
int main(void) {  
    double x = 2.5, y;  
    y = cube(x);  
    printf("%f %f", x, y);  
    return 0;  
}
```



# زبان برنامه نویسی C (21814)

## How Do Functions Work

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

```
double cube(double val);
```

// function prototype (declaration)

```
int main(void) {
```

```
    double x=2.5, y;
```

// 1

```
    y = cube(x);
```

// 2 function call

```
    printf("%f %f", x, y);
```

// 3

```
    fflush(stdout);
```

// 4 forces output to console screen

```
    return 0;
```

// 5

```
}
```

input to function

```
double cube(double val){
```

// function definition

```
    return val*val*val;
```

// 6

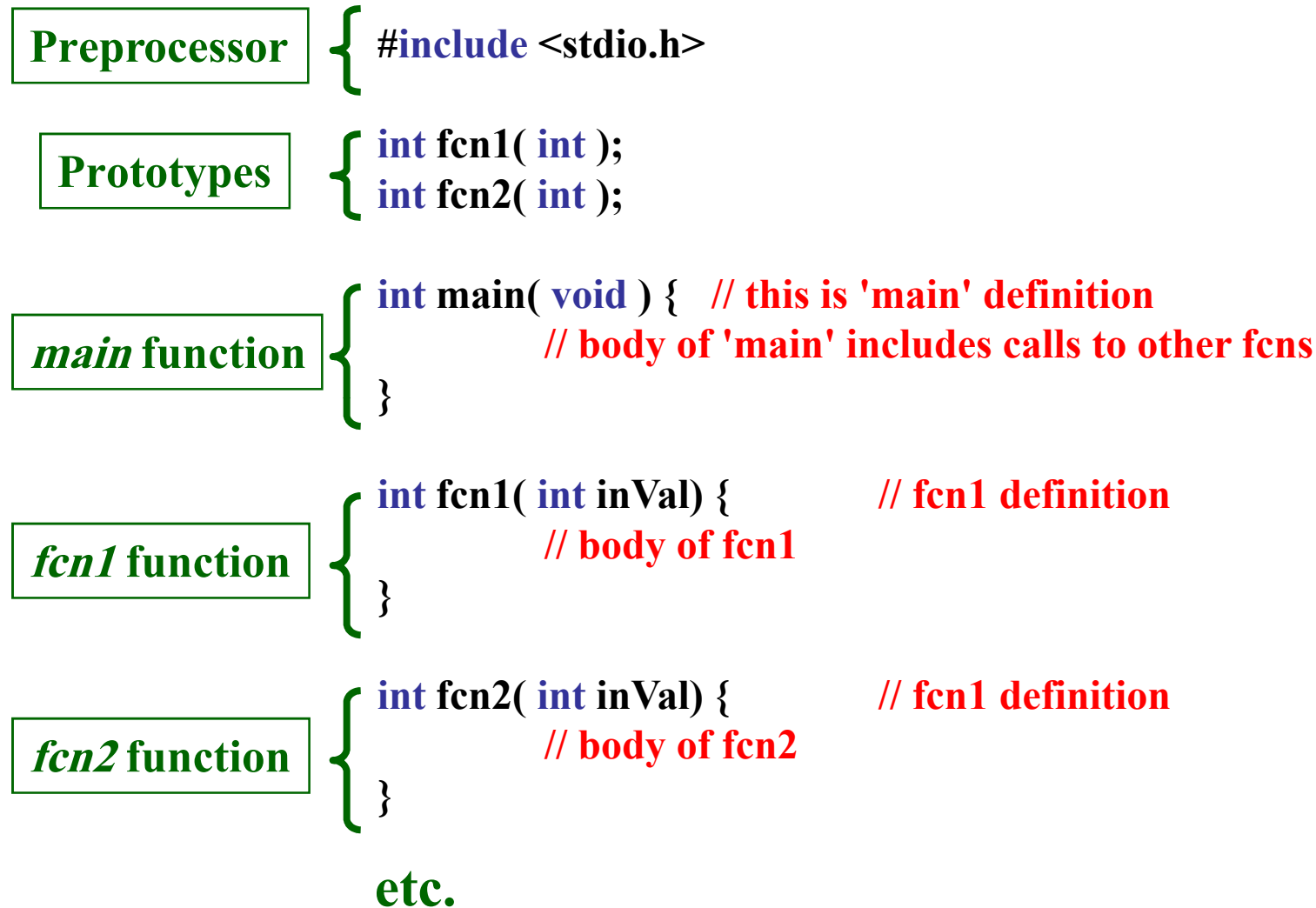
```
}
```

output from function

Execution order is:

1 2 6 2 3 4 5

# Program Structure – Big Picture.



# زبان برنامه نویسی C (21814)

```
#include <stdio.h>           // libraries
```

```
int square(int a); // prototypes  
int cube(int a);
```

```
int main(void) { // main function  
    int i = 1;
```

```
    while(i < 10) {  
        printf("%5d %5d %5d\n", i, square(i), cube(i)); // function calls  
        ++i;  
    }  
    return 0;
```

```
}
```

```
int square(int a) { // square function  
    return a * a;  
}
```

```
int cube(int a) { // cube function  
    return a * a * a;  
}
```

## Program Style 1 (this course)

## زبان برنامه نویسی C (21814)

```
#include <stdio.h> // libraries
```

```
int square(int a) { // square function  
    return a * a;  
}
```

```
int cube(int a) { // cube function  
    return a * a * a;  
}
```

```
int main(void) { // main function  
    int i = 1;  
  
    while(i < 10) {  
        printf("%5d %5d %5d\n", i, square(i), cube(i));  
        ++i;  
    }  
    return 0;  
}
```

## Program Style 2 (no prototypes)

## Function Call Details

```
int main(void) {  
    double y, x = 2.5;  
    y = cube(x);  
    return 0;  
}
```

```
double cube(double val) {  
    return val*val*val;  
}
```

1) store value of *x* in a temporary location, jump to function *cube()*

2) assign value of *x* to *val*

3) calculate cube of *val* and store in temporary place and jump back to calling line.

4) retrieve the cube value and assign to *y*.



## زبان برنامه نویسی C (21814)

### صدا زدن یک تابع با Array Argument (صفحه 383 کتاب)

صدا زدن یک تابع با یک argument بصورت آرایه بصورت زیر انجام می شود:

— هنگان صدا زدن تابع، تنها نام آرایه بدون براکت ها، یا یکی از اعضا و یا طول آن ذکر می شود.

— همچنین نیازی نیست که اپراتور & قبل از نام آرایه ذکر شود.

— در هنگان declare نمودن باید براکت ها را بدون ذکر طول آرایه نمایش داد.

— در این حالت اگر طول آرایه ذکر شود مانعی ندارد ولی کامپایلر آن را در نظر نمی گیرد.

— در این حالت می توان از تابع با آرایه هایی با طول متفاوت استفاده نمود.

## زبان برنامه نویسی C (21814)

صدا زدن یک تابع با **Array Argument** (ادامه)  
مثال:

```
void Fname (double Arrayname[], int n);
```

```
void getdata (double x[], int n);
```

## زبان برنامه نویسی C (21814)

```
#include "tools.h"

#define N    50                /* Maximum number of data values. */

void  get_data( double x[], int n );
double average ( double x[], int n );
double variance( double x[], int n, double mean );

void main( void )
{
    double x[N];                /* Array for data values. */

    int num;                    /* Actual number of data values. */
    double mean;                /* The mean of the values in array x. */
    double var;                 /* Variance of the data in array x. */
    double stdev;               /* Standard deviation of the data in array x. */
    banner();

    printf( "\n Enter number of values in data set (2..%i): ", N );
    for(;;) {
        scanf( "%i", &num );
        if (num > 1 && num <= N) break;
        printf( "Error: %i is out of legal range, try again: \n", num );
    }
    printf( " Computing statistics on %i data values.\n", num );

    get_data( x, num );
    mean = average( x, num );
    var = variance( x, num, mean );
    stdev = sqrt( var );

    printf( "\n The mean of the %i data values is = %.2f \n", num, mean );
    printf( " The variance is = %.2f \n", var );
    printf( " The standard deviation is = %.2f \n", stdev );
    bye();
}
```

## زبان برنامه نویسی C (21814)

```
/* ----- */
/* Given an empty array of length n, input data values to fill it. */
void
get_data( double x[], int n )
{
    int k;                                /* Loop counter and subscript. */
    puts( "Please enter data values when prompted." );
    for (k = 0; k < n; ++k) {
        printf( "x[%i] = ", k );        /* Prompt for kth value. */
        scanf( "%lg", &x[k] );          /* Read into array slot k. */
    }
}
```

## زبان برنامه نویسی C (21814)

```
/* -----  
** Given an array of n values, print the values and calculate the mean.  
*/  
double  
average( double x[], int n )  
{  
    double sum;  
    int k;                                /* Loop counter and subscript. */  
    for (sum = k = 0; k < n; ++k) {      /* Echo and sum the k values. */  
        printf( "\n      x[%i] = %.2f", k, x[k] );  
        sum += x[k];  
    }  
    return sum / n;  
}
```

```
/* -----  
** Given an array of values and their mean, calculate the variance.  
*/  
double  
variance( double x[], int n, double mean )  
{  
    double divisor, sum;  
    int k;  
    for (sum = k = 0; k < n; ++k) {      /* Compute and sum the k squares. */  
        sum += pow( (x[k] - mean), 2 );  
    }  
    if (n < 20) divisor = n - 1;  
    else divisor = n;  
    return sum / divisor;  
}
```

## زبان برنامه نویسی C (21814)

### Recursive Function (Type I):

```
#include <stdio.h>
int a,N=1;
int Factoriel(int x);
void main (void){
    printf(" Enter a number:");
    scanf("%i",&a);
    Factoriel(a);
    printf("\nThe factoriel of %i is equal to %i.\n\n",a,N);
}
int Factoriel(int a){
    N = N*a;
    a=a-1;
    if(a>1){
        Factoriel(a);
    }
    return 0;
}
```

## زبان برنامه نویسی C (21814)

### Recursive Function (Type II):

```
#include <stdio.h>
int a,N=1,W=1;
int Factoriel(int x);

void main (void){
    printf(" Enter a number:");
    scanf("%i",&a);
    N= Factoriel(a);
    printf("\nThe factoriel of %i is equal to %i.\n\n",a,N);
}
int Factoriel(int a){
    W = W*a;
    a=a-1;
    if(a>1){
        Factoriel(a);
    }
    return W;
}
```

## زبان برنامه نویسی C (21814)

### تمرین 5:

مثال ریختن دو تاس منصف را در نظر بگیرید. اگر متغیر تصادفی  $X$  بیانگر مجموع نتایج دو تاس باشد، مطلوب است محاسبه میانگین، انحراف معیار، و واریانس متغیر تصادفی  $X$ . در مورد این مثال میانه و مد کدامند؟

این آزمایش را برای 10، 100، 1000 و 10000 مرتبه انجام دهید و نتایج را در یک فایل خروجی ذخیره کنید.





بنام خدا

زبان برنامه نویسی C (21814)

نیمسال اول تحصیلی 1386-1387

Lecture 9

Selected Topics

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814)

نکاتی برگزیده از زبان برنامه نویسی C:

- روشی دیگر برای ارسال متن به روی خروجی:

```
printf ("%s \n %s", "one", "two");
```

which prints:

```
one  
two
```

## زبان برنامه نویسی C (21814)

Integers and strings can also use field width  
%nd, %ns where n is again min field width.

**Example: printf("%-10d%10s", 21,"abcde")**

21 8 wide 5 wide abcde  
10 wide 10 wide

## زبان برنامه نویسی C (21814)

مثال:

```
/*  
 * Name: Nasser Salmasi  
 * Date: 10/3/06  
 * Assignment: Term Project1  
 * Description: Johnson's Algorithm.  
 */  
  
#include <stdio.h>  
  
int main(void) {  
    /* print the message */  
    printf("Whatever you want!");  
    return 0;  
}
```

**File Header Comment**

**Blank Lines (readability)**

**Inline Comment**

**Indenting – Use Tab Key!**

## زبان برنامه نویسی C ( 21814 )

**مثال:**

- **Print the squares and cubes of integers from 1 through 3.**
- **Declare variables needed.**
- **Calculate the numbers.**
- **Print the results.**

# The Complete Program

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

```
int main(void) {
```

```
    int n, square, cube;    /* n is input, square & cube are output */
```

```
    printf("Table of squares and cubes\n");
```

```
    n = 1;
```

```
    square = n * n; cube = n * square; /* first line of output */
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    n = 2;
```

 **note**

```
    square = n * n; cube = n * square; /* second line of output */
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    n = 3;
```

```
    square = n * n; cube = n * square; /* third line of output */
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    return 0;
```

```
}
```

```
/*  
 * Name: Nasser Salmasi  
 * * Date: Oct 6, 2006  
 * Assignment: Squares Cubes  
 * Description: Make a table of the integers from 1 – 3 and their squares and cubes.  
 */
```

header comment

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

readability  
(blank lines)

```
int main(void) {
```

inline comments

```
    int n, square, cube;
```

// n is input, square & cube are output

```
    printf("Table of squares and cubes\n");
```

```
    n = 1;
```

```
    square = n * n; cube = n * square; // first line of output
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    n = 2;
```

```
    square = n * n; cube = n * square; // second line of output
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    n = 3;
```

```
    square = n * n; cube = n * square; // third line of output
```

```
    printf("%d %d %d\n", n, square, cube);
```

```
    return 0;
```

indentation -- Use Tab Key!

```
}
```

## زبان برنامه نویسی C (21814)

### Same Program in Unreadable Form

```
#include <stdio.h>
int main(void) { int n, square, cube;
printf("Table of squares and cubes\n");
n = 1; square = n * n; cube = n * square;
printf("%d %d %d\n", n, square, cube); n = 2;
square = n * n; cube = n * square;
printf("%d %d %d\n", n, square, cube); n = 3;
square = n * n; cube = n * square;
printf("%d %d %d\n", n, square, cube);
return 0; }
```



## زبان برنامه نویسی C (21814)

### Precedence

- Precedence = rank or priority.
- Some operators have priority over others (i.e should be done first)  
 $1 + 2 * 3$  same as  $1 + (2 * 3)$  (i.e. = 7)  
i.e.  $*$  has higher precedence than binary  $+$  or  $-$
- Parentheses force precedence.  
 $(1 + 2) * 3$  is equal to 9

## زبان برنامه نویسی C (21814)

### Associativity

- **When operators have same precedence we must use rules of associativity (left to right or right to left)**
- **+ - same precedence, left to right assoc.**  
 **$1+2+3-4+5$  means  $((1+2)+3)-4+5 = 7$**
- **\* / % same precedence, left to right assoc.**  
 **$5 / 2 * 2$  means  $(5/2)*2 = 4$  **not**  $5/(2*2) = 1$**

## زبان برنامه نویسی C (21814)

### Precedence /Associatively Table

<u>operation</u>	<u>precedence</u>	<u>associatively</u>
( )	1 (highest)	L to R
+(plus) -(minus)	2	R to L
* / %	3	L to R
+ -	4	L to R
= (assignment)	14 (lowest)	R to L

## زبان برنامه نویسی C (21814)

### **Use Parentheses!!!!!!!**

- **Best to force precedence and associatively with parentheses.**
- **Easier to read**
- **Everyone gets confused about these rules!**

## زبان برنامه نویسی C (21814)

### Increment, Decrement Examples

Let  $b = 2$  and  $c = 3$ .

- After assignment  $a = (++b) + (++c);$   
the values will be  $a=7, b=3, c=4$
- After the assignment  $a = (b++) + (c++);$   
the values will be  $a=5, b=3, c=4$
- $--i;$  is a perfectly good statement.  
Same as  $i = i - 1;$

## زبان برنامه نویسی C (21814)

### Examples - Expressions

- If  $c = 0$ , then what are values of  $a$ ,  $b$ ,  $c$  after these two statements are executed.

$a = ++c;$

$b = a++;$

$a = \quad , b = \quad , c =$

- Avoid expressions like  $a = c + ++c;$  where a variable and its increment or decrement occur in same expression (see the code).

## زبان برنامه نویسی C (21814)

### Assignment Operator

- **An assignment has a value!**  
**a = (b = 0); same as a=0; b=0;**  
**or a = b = 0; (because of right to left associatively).**

## زبان برنامه نویسی C (21814)

```
int a = -2, b = 3, c = 4, d = 5;
```

```
d *= a-- + ++b * c - c % 2;
```

- What is d after executing this statement?
- But, **please don't write like this!**



## زبان برنامه نویسی C (21814)

### Find the 8 Programming Errors

```
#include <stdio>;
#define TOPBOTTOM "III";
# define MIDDLE "\nI"

void main(void) {
    printf("/n/n/n");
    printf(TOPBOTTOM)
    printf(MIDDLE); printf(MIDDLE);
    printf(\n);
    prinf(TOPBOTTOM);
}
```

## زبان برنامه نویسی C (21814)

### نحوه تعریف متغیر برای ذخیره کاراکتر

- برای معرفی متغیر به منظور ذخیره کاراکتر می توان بصورت زیر عمل نمود:

```
char b = 'z';
```

- در این حالت متغیر b می تواند یک کاراکتر 0-9, a-z, A-Z و غیره را ذخیره کند. در واقع در این حالت فقط یک بایت برای کاراکتر b اختصاص می یابد.

- می توان آرایه برای ذخیره کاراکتر نیز تعریف نمود (مثال):

```
char Name[20];
```

## زبان برنامه نویسی C (21814)

**نحوه تعریف متغیر برای ذخیره کاراکتر و ارسال به خروجی:**

- نحوه گرفتن کاراکتر از کاربر و همچنین ارسال آن به خروجی بصورت زیر است:

```
char b = 'z';  
b = getchar();  
printf( “ %c”, b);  
putchar (b);
```

## Some other useful functions

These all require **#include <ctype.h>**

- **isalpha(c)** - returns nonzero if c is letter
- **isdigit(c)** - returns nonzero if c is a digit (0-9)
- **isspace(c)** - returns nonzero if c is space, tab or newline.
- **islower(c)** - returns nonzero if c is lower case.
- **isupper(c)** - returns nonzero if c is upper case.
- **toupper(c)** – returns upper case char.
- **tolower(c)** – returns lower case char.

## زبان برنامه نویسی C (21814)

برنامه زیر را در نظر بگیرید:

```
int i;
```

```
i = 2147483647;
```

```
printf("%d %d %d", i, i+1, i*i);
```

فکر می کنید نتیجه چه باشد؟

## What's going on here?

- **Computers store only so many digits.**
- **When an operation creates more than they have, the higher digits are lost.**
- **Just like 5-digit odometer on a car.**
  - **What comes after 99,999? -- 00,000 right?**
- **Same in computers -- next integer after the largest integer is the smallest integer**

## زبان برنامه نویسی C (21814)

**نحوه بازکردن یک فایل در زبان برنامه نویسی C:**

**خصوصیت های یک فایل:**

- هر فایل دارای یک نام
- هر فایل دارای اندازه (Size) (بر حسب کیلو بایت)
- هر فایل شامل انواع مختلفی از داده ها
- هر فایل را می توان بصورت اطلاعاتی که در یک دیسک جانبی ذخیره شده اند تعبیر نمود.

## زبان برنامه نویسی C (21814)

```
FILE* INDATA; // *fp means fp points to a FILE structure
INDATA=fopen("Data1.txt","r");
fscanf(INDATA," %i\n",&Machine_Number);
for(f1=1;f1<=group_number;f1++){
    for(f2=1;f2<=job_number[f1];f2++){
        for(f3=1;f3<=Machine_Number;f3++){
            fscanf(INDATA," %i" ,&time_M[f1][f2][f3]);
        }
    }
    fscanf(INDATA,"\n");
}
fclose(INDATA);
```



## زبان برنامه نویسی C (21814) نحوه خواندن یک فایل

`fopen(filename, mode)`

`int a, b, c;`

`FILE *fp; // *fp means fp points to a FILE structure`

`fp = fopen("data.txt", "r");`

`fscanf(filepointer, string, &var1, &var2, ...)`

`// read three integers from the file`

`fscanf(fp, "%d %d %d", &a, &b, &c);`

`fclose(filepointer)`

`fclose(fp); // close the file`

## زبان برنامه نویسی C (21814) نحوه خواندن یک فایل

test for equality

### Open Errors

- If `fp == NULL`, there was an error and your program cannot proceed!
- Possible causes:
  - filename doesn't exist in current folder.
  - file is being used (i.e. is opened) by another application.
  - file is corrupted (i.e has been damaged).

## زبان برنامه نویسی C (21814) نحوه خواندن یک فایل

### File Mode - text files

- ***mode* can be "r" "w", "a"**
- **"r" open a text file for reading from the beginning (fscanf)**
- **"w" open a text file for writing from the beginning (fprintf). Replaces any existing file or creates new one.**
- **"a" open a text file for writing starting at the end (fprintf).**

## زبان برنامه نویسی C (21814) نحوه خواندن یک فایل

- In text files, numbers, characters, and strings are all stored as text characters.
  - 'a' stored as the character *a*.
  - "a string" stored as the characters *a string*.
  - 1234567890 (a 4 byte int) stored as the 10 ASCII characters *1234567890*.
  - 123.456 (a 4 byte float) stored as the 7 ASCII characters *123.456*.
- White space is used to separate.

## زبان برنامه نویسی C (21814) نحوه خواندن یک فایل

- `fscanf(filepointer, formatString, var, var,...)`

```
FILE *fp;  
int a, c;  
double b;  
fp = fopen("input.txt", "r");  
fscanf(fp, "%d %lf %d", &a, &b, &c);  
printf("%d\n%f\n%d", a, b, c);  
fclose(fp);
```

white space



input.txt

234	5.345
567	

- 234, 5.345, and 567 will be assigned to a, b, and c respectively

## How scanf (or fscanf) Reads

- **When using %d, %f, or %lf, scanf:**
  - 1. skips white space**
  - 2. reads the ASCII digits (and/or '+', '-', '.') until it encounters white space.**
  - 3. converts the digits to an integer, float, or double**
- **Returns number of successful conversions or EOF (if encountered an error).**

## زبان برنامه نویسی C (21814)

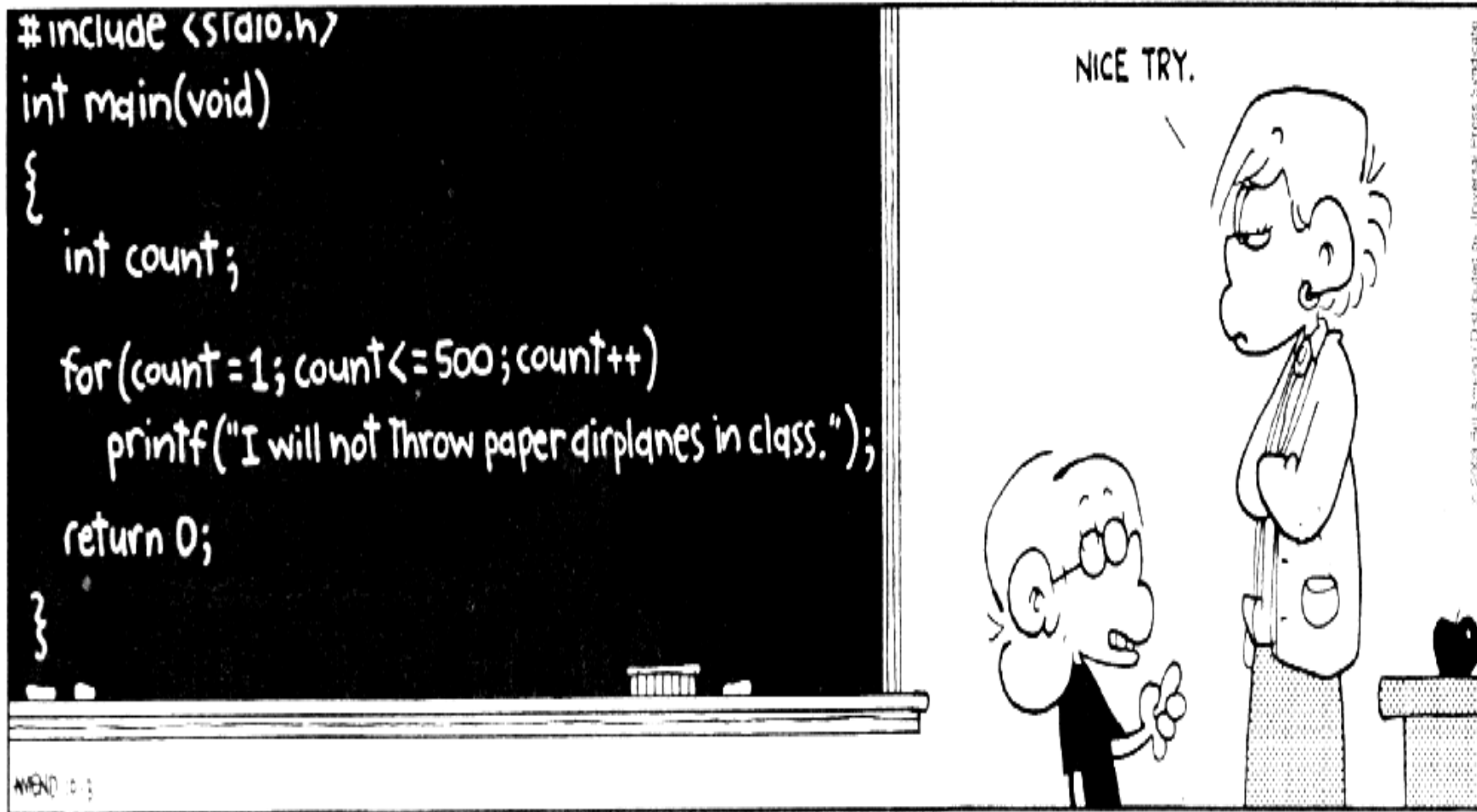
### نحوه نوشتن به روی فایل:

- برای نوشتن داخل یک فایل از دستور `fprintf` استفاده می کنیم. نحوه استفاده از این دستور همانند دستور `printf` است.

مراجعه به مثال

# Foxtrot Version

FOXTROT — By Bill Amend





## زبان برنامه نویسی C (21814)

### تمرین:

برنامه ای بنویسید که تعداد دانشجویان ،نام دانشجو، شماره دانشجویی، نمرات میان ترم و پایان ترم دانشجویان کلاس را از فایل HW.txt گرفته، میانگین و واریانس نمرات میان ترم و پایان ترم را محاسبه کند. همچنین برای هر دانشجو اختلاف نمرات میان ترم و پایان ترم وی را با میانگین گزارش دهد و نتایج را در فایلی به نام Report.txt گزارش کند.

زبان برنامه نویسی C (21814)

## Bubble sort

- Sort an array of numbers or characters into numeric or alphabetic order.
- 1) Compare last two -- swap, if out of order
  - 2) Compare next to last two -- etc.
  - 3) Start again at end but stop one shy of beginning
  - 4) Continue

## زبان برنامه نویسی C (21814)

### Bubble sort first pass

4	6	1	8	2	9	start
4	6	1	8	2	9	loop1
4	6	1	2	8	9	loop2
4	6	1	2	8	9	loop3
4	1	6	2	8	9	loop4
1	4	6	2	8	9	end pass 1

زبان برنامه نویسی C (21814)

## Bubble sort -- second pass

1	4	6	2	8	9	start pass two
1	4	6	2	8	9	loop1
1	4	6	2	8	9	loop2
1	4	2	6	8	9	loop3
1	2	4	6	8	9	end of pass two

زبان برنامه نویسی C (21814)

## Bubble sort - 3rd pass

1 2 4 6 8 9      start pass three

1 2 4 6 8 9      loop1

1 2 4 6 8 9      loop2

1 2 4 6 8 9      end of pass three

(Note: nothing actually done)

زبان برنامه نویسی C (21814)

## Bubble sort - 4rd pass

1 2 4 6 8 9      start pass four

1 2 4 6 8 9      loop1

1 2 4 6 8 9      end of pass four

زبان برنامه نویسی C (21814)

## Bubble sort - 5th pass

1 2 4 6 8 9      start of pass 5

1 2 4 6 8 9      done

- Note that this computer algorithm is quite different from the way we do this by hand.
- Another example of the difficulty of doing something with a computer that is easy with human brain.

## زبان برنامه نویسی C (21814)

### Bubble sort - code

```
void bubble(int a[], int n) {  
    int i, j;  
    // i counts passes, j cycles thru swap positions  
    // numbers are: a[0], a[1], ... a[n-2], a[n-1]  
    for(i=0; i < n-1; ++i)  
        for(j=n-1; j > i; --j)  
            if( a[j-1] > a[j]) // if out of order, swap  
                swap(&a[j-1], &a[j]);  
    return;  
}
```



## زبان برنامه نویسی C (21814)

### Bubble Sort Analysis

- Assume  $N$  items to sort.
- 1st pass  $N-1$  loops.
- 2nd pass  $N-2$  loops.
- Last pass 1 loop.
- $(N-1) + (N-2) + \dots + 1 = N*(N-1)/2$
- Execution time increases as square of size, which is not as good as QuickSort,  $N*\ln(N)$ .



بنام خدا

زبان برنامه نویسی C (21814)

Lecture 10

Chapter 11

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814)

### تمرین:

همانطور که می دانید به اعدادی اعداد فیثاغورثی می گویند که رابطه زیر میان آنها برقرار باشد:

$$a^2=b^2+c^2$$

برنامه ای بنویسید که تمامی رشته های 3 تایی اعداد فیثاغورثی را بیابد و به عنوان خروجی چاپ کند.

## زبان برنامه نویسی C (21814)

```
#include <stdio.h>
#define N 20

int main(void) {
    int i,j,k, sumSquares;

    for (i=1; i <=N; ++i) { /*loop through all i's and j's <= N */
        for (j=1; j <= N; ++j) {
            sumSquares = i*i + j*j;
            for(k=1; k <=N; ++k) /* see if sum is perfect square */

                if (sumSquares == k*k)
                    printf("%5d %5d %5d\n", i, j, k);
        }
    }
    return 0;
}
```

## زبان برنامه نویسی C (21814) فصل یازدهم

- در این فصل **character data type** ها مورد بررسی قرار می گیرند.

## زبان برنامه نویسی C (21814) فصل یازدهم

روش های مقدار دهی به یک متغیر **char**:

توابع **standard library** برای دریافت اطلاعات و ذخیره در یک متغیر **char** عبارتند از:

1. **getchar()**: این تابع هیچ پارامتری ندارد و می تواند یک کاراکتر را بخواند و بازگرداند.  
**Ch = getchar();**

2. **scanf()** بصورت **"%c"**: در این روش برنامه اولین ورودی را خوانده بدون توجه به اینکه متغیر بعدی " " باشد یا نه. در واقع این روش دقیقا معادل تابع **getchar()** است.  
**scanf("%c",&Ch);**

## زبان برنامه نویسی C (21814) فصل یازدهم

روش های مقدار دهی به یک متغیر char(ادامه):

3. **scanf()** بصورت **"%c"** در این روش فضاهاى " " هنگام خواندن مد نظر قرار نمی گیرند و اولین کاراکتر ثبت شده دریافت می شود.

```
scanf(" %c", &Ch);
```

- با توجه به این مطلب همواره استفاده از روش سوم قویا توصیه می شود.

## زبان برنامه نویسی C (21814) فصل یازدهم

مثال:

```
#include <stdio.h>

void main (void){

    char d;
    printf(" Enter a character:");
    d=getchar();

    printf(" %c\n\n",d);

    printf(" Enter aother:");
    scanf(" %c",&d);
    printf(" %c\n\n",d);

}
```



## زبان برنامه نویسی C (21814) فصل یازدهم

### روش های مقدار دهی به یک متغیر **char** (ادامه):

- هنگامی که اطلاعات توسط کاربر وارد می شود، اطلاعات وارد شده تا قبل از اینکه کاربر دگمه **enter** را فشار ندهد وارد برنامه نمی شود و در فضایی به نام **keyboard buffer** ذخیره می شود.
- به محض فشار دادن دگمه **enter**، مقادیر تایپ شده به مکان دیگری به نام **input buffer** منتقل می شوند و توسط برنامه قابل دسترسی می شوند.

## زبان برنامه نویسی C (21814) فصل یازدهم

### روش های مقدار دهی به یک متغیر char(ادامه):

- در این حالت برنامه تعداد کاراکتر مورد تقاضا را از ورودی می گیرد. ورودی های اضافی وارد شده در این مکان باقی می ماند و در مرحله بعدی که برنامه در انتظار مقادیر ورودی از کاربر است (توسط دستور های printf و یا scanf) به برنامه منتقل می شوند.

## زبان برنامه نویسی C (21814) فصل یازدهم

نحوه ارسال مقادیر متغیر **char** به روی خروجی:

دو روش اصلی در این زمینه عبارتند از:

- **putchar()**: در صورتی که نیاز به ارسال یک کاراکتر باشد استفاده می شود.
- **printf()**: بهترین روش برای ارسال مقادیر

## زبان برنامه نویسی C (21814)

### نحوه غلط یابی (Debugging):

- همواره پس از نوشتن چند خط از صحت برنامه اطمینان حاصل کنید.
- از مثال های ساده استفاده کنید.
- در صورت وجود اشتباه سعی کنید مکان واقعی اشتباه را پیدا کنید.

## زبان برنامه نویسی C (21814)

### Function Libraries

- **<stdio.h>** - printf(), fprintf(), scanf(), fscanf(), fopen(), putchar(), getchar(), etc.
- **<math.h>** - pow(), sqrt(), fabs(), etc.
- **<ctype.h>** - toupper(), tolower(), isalpha(), isdigit(), etc.
- **<stdlib.h>** - rand(), srand(), exit(), etc.
- **<time.h>** - time(), clock(), etc.

## زبان برنامه نویسی C (21814)

### <stdlib.h>

- **exit(val)** example  
if (fp == NULL)  
exit(1); // **e.g. quit program if error in fopen**
- **rand(void)** example  
for(i=0; i < 100; ++i) // **100 random numbers**  
printf("%d ", rand());
- **srand(value)** example  
srand(100); // **sets beginning of sequence**

## زبان برنامه نویسی C (21814)

### The rand() Function

- **Use:**  
`i = rand();`    *//requires #include <stdlib.h>*
- **Returns different number each time it's called (between 0 and RAND\_MAX)**
- **RAND\_MAX = 32,767 in our current software.**
- **Always produces same sequence.**  
**16838 5758 10113 17515 etc.**

## زبان برنامه نویسی C (21814)

### The srand() function

- Sets starting point for random sequence, e.g.  
`srand (seed);`
- 'seed' will determine the sequence `rand()` returns.
- Also in `<stdlib.h>` library



## زبان برنامه نویسی C (21814)

### The time() Function

- Returns the number of seconds since 1/1/1970.
- At 7:30 pm, 1/29/2006, the value was 1,138,563,157
- Usage: (requires #include <time.h>)  
`int t;`  
`t = time(NULL);`



null pointer

## زبان برنامه نویسی C (21814)

### نحوه ثبت زمان اجرای یک برنامه

# Tick Tock

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

int main(void) {
    int i, t;
    for(i=0; i <= 10; ++i) {           // loop for ticks and tocks
        t = time(NULL) + 1;           // t is 1 second from now
        while (time(NULL) < t) // wait */
            ;
        printf("TICK\n");              // then print TICK
        t = time(NULL) + 1;           // again 1 sec in future
        while(time(NULL) < t) // wait again
            ;
        printf("TOCK\n");              // print TOCK and repeat
    }
    return 0;
}
```

## زبان برنامه نویسی C (21814)

### تمرین کلاسی: مساله حرکت تصادفی (Random Walk)

فرض کنید شخص A برای ملاقات شخص B به مکانی رفته است. شخص B بنا به دلایلی تاخیر دارد، لذا شخص A تصمیم می گیرد که برای وقت گذرانی در ابتدای هر دقیقه سکه ای را پرتاب کند و در صورت شیر آمدن یک قدم به عقب بردارد. محاسبه کنید که در صورت تاخیر شخص B به میزان یک ساعت، شخص A در چه فاصله ای قرار خواهد داشت؟ مساله را در حالت های مختلف اینکه احتمال شیر آمدن برابر مقادیر زیر P باشد محاسبه و نتایج را در یک فایل خروجی ارسال کنید:

$$P = 0.1i \quad i = 0, 1, 2, \dots, 9, 10$$



بنام خدا

زبان برنامه نویسی C (21814)

Lecture 11

Chapter 12: Pointers

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814)

### فصل دوازدهم: مقدمه ای بر Pointerها

- مروری بر آخرین بخش با قیمانده از انواع ابتدایی data type ها، یعنی Pointer ها

- وظیفه Pointer اشاره به آدرس یک data object در حافظه است.

- در این فصل نحوه صحیح بکار گیری Pointerها و سه اپراتور اصلی آنها، یعنی =, \*, & مورد بررسی قرار می گیرند.

## زبان برنامه نویسی C (21814) Pointers

- با بکار گیری Pointer Parameters می توان بیش از یک نتیجه از یک function گرفت.
- Pointer نقشی همانند ضمیر در زبان محاوره دارد و می تواند در هر لحظه به شی متفاوتی اشاره کند. دلایل بکار گیری Pointer ها در زبان C عبارتند از:
  - دریافت بیش از یک مقدار از یک function
  - ایجاد و پردازش stringها
  - کنترل بهتر به روی آرایه ها
  - ایجاد data structure هایی که اندازه آنها تغییر می کند.

## زبان برنامه نویسی C (21814) Pointers

- استفاده از Pointer در زبان C موجب افزایش انعطاف در برنامه نویسی می شود.

- اگر p یک متغیر pointer باشد و آدرس متغیر k در آن ذخیره شده باشد، می توان گفت که p به k اشاره می کند.

- p points at k
- p is reference to k
- p refers indirectly to k



## زبان برنامه نویسی C (21814) Pointers

### نحوه تعریف متغیرهای pointer:

- برای تعریف متغیر pointer، ابتدا نوع متغیر هایی که pointer تعریف شده می تواند به آنها اشاره کند ذکر می شود.
- سپس نام pointer بصورت زیر ذکر می شود:

**int** \*p1, \*p2;

- استفاده از علامت \* قبل از p1 الزامی است.
- هر pointer فقط می تواند به نوع متغیر هایی که تعریف شده اشاره کند.

## زبان برنامه نویسی C (21814) Pointers

### مقدار دهی ابتدایی به pointer ها

- اگر یک pointer بدون مقدار دهی اولیه تعریف شود، حافظه ای به آن تخصیص می یابد بدون اینکه آدرسی در آن ذخیره شود، لذا pointer به هر چه در قبل در آن ذخیره شده اشاره خواهد نمود.
- توجه داشته باشید که همواره یک pointer به جایی اشاره می کند.
- مقدار اولیه pointer می تواند اشاره به جایی بی معنی در برنامه جاری یا مکانی تصادفی در متن برنامه باشد.

## زبان برنامه نویسی C (21814) Pointers

### مقدار دهی ابتدایی به pointer ها (ادامه)

- اغلب کامپایلر های C پیغام **ERROR** در مورد مقدار اولیه یک **pointer** ارسال نمی کنند.
- در چنین مواقعی هنگام اجرای برنامه بر اساس مقدار اولیه ذخیره شده، اتفاقات مختلفی ممکن است روی دهد.
- برای جلوگیری از این مشکل بهتر است از **NULL pointer** استفاده شود.

## زبان برنامه نویسی C (21814) Pointers

- **NULL pointer** در **stdio.h** تعریف شده و نقش همان مقدار دهی اولیه مساوی صفر برای متغیرها را دارد.
- اگر مقدار **NULL** در **pointer** ذخیره شود، آنگاه **pointer** به هیچ جا اشاره نمی کند (در واقع به **memory Location 0** اشاره می کند).

```
int *pt = NULL;
```

## زبان برنامه نویسی C (21814) Pointers

- یک متغیر `pointer` مکانی است که یک `pointer` در آن ذخیره می شود.
- همانطور که ذکر شد، `pointer` خودش آدرس یک متغیر دیگر را در خود ذخیره می کند. بصورت صریح تر: یک متغیر `pointer` شامل یک آدرس است در حالی که خود متغیر آدرس خودش را دارد.
- گاهی اوقات نقش `pointer` ها حتی برای برنامه نویسان حرفه ای گیج کننده است (صفحه 445 کتاب).

## زبان برنامه نویسی C (21814) Pointers

- همانطور که قبلا دیده شد،  $\&k$  آدرس متغیر  $k$  را نشان می دهد.

- اپراتور  $*$  ، که  $\text{indirection}$  نامیده می شود، عملی مخالف اپراتور  $\&$  را انجام می دهد.

- در واقع  $*p$  به معنی مرجع  $p$  (متغیری که  $p$  به آن اشاره می کند) می باشد. لذا:

$*\&k == k$

$*\&p == p$

## زبان برنامه نویسی C (21814) Pointers

- عبارت  $*p$  دقیقاً همان کاری را انجام می دهد که یک ضمیر با اسم انجام می دهد.
- اگر  $p$  آدرس متغیر  $k$  را ذخیره کرده باشد، آنگاه:

$$m = *p \sim m = k$$

$$*p = n \sim k = n$$

- در واقع هر جا که می خواهیم نام متغیری را بنویسیم، می توانیم از یک `pointer` که به آن متغیر اشاره می کند استفاده کنیم.

## زبان برنامه نویسی C (21814) Pointers

مثال:

```
int k, m;
```

```
int *p;
```

```
p = &k;
```

```
m = *p + 2; ~ m = k + 2;
```



## زبان برنامه نویسی C (21814) Pointers

### نحوه تخصیص یک pointer:

- سه روش کلی برای تخصیص pointer وجود دارد:

1. تخصیص صریح:  $p = \&k;$

2. دسترسی به یک مقدار توسط یک pointer و اختصاص مقدار آن به یک متغیر  
 $m = *p;$

3. مقدار دهی غیر مستقیم به متغیری که pointer به آن اشاره می کند.  
 $*p = m+2;$

## زبان برنامه نویسی C (21814) Pointers

مثال:

```
#include <stdio.h>
void main(void) {
    int *pt=NULL;
    int *p;
    int k=17,m;
    p=&k;
    pt=p;
    m=*p +2;
    *p = m;
    printf( " address of p: %i contents of p: %i\n", (int) &p, (int)p);
    printf( " address of pt: %i contents of pt: %i\n", (int) &pt, (int)pt);
    printf( " address of k: %i contents of k: %i\n", (int) &k, (int)k);
    printf( " address of m: %i contents of m: %i\n", (int) &m, (int)m);
    printf( "\n\n");
}
```

## زبان برنامه نویسی C (21814) Pointers

- Yet another use for **&** and **\***
- **int** *year* = 5; *year* represents the value stored in a certain location (address) in memory.
- The **memory address**, i.e., the address of *year*, is called a pointer to *year* -- it gives the location of *year*, not its value.

## The 'address of' operator, &

- Given **int** years; then **&years** is the memory address of this variable.
- ***&years*** is a constant pointer.
- ***scanf("%d", &years)*** – **scanf()** needs the address of where to store the data.

زبان برنامه نویسی C (21814)

## The de-reference operator \*

- An address of a variable can be called a *reference* to it.
- The \* unary operator is the inverse of &.
- It gets the thing referred to. It 'de-references' the reference.
- **\*&a** is just the value of *a*

زبان برنامه نویسی C (21814)

## Pointer variables

- A variable that can hold an address is called a pointer variable.
- An integer pointer variable is declared as: **int \***  
**p;**
- **int \*** is a new type called *pointer to int*
- **p** is a variable of type *pointer to int*.
- Only addresses of integers can be stored in **p**.

## More Syntax

```
int *p, years, a;  
years = 20;  
p = &years;      // p holds address of years  
a = *p;          // get value stored at p & store in a  
*p = 30;          // years changed to 30  
printf("%d %d %d %d", years, a, *p, p);
```

Output:

30 20 30 1245000

## زبان برنامه نویسی C (21814)

### \* on left vs. \* on right of =

- \*p on right side of assignment means:
  - "get the value to which p points"
  - ex: `a = *p;`
- \*p on left side of assignment means:
  - "store the value of assignment in the location pointed to by p"
  - ex: `*p = 30;`



## زبان برنامه نویسی C (21814)

### What's it look like in memory

```
int years, *p;  
years = 20;  
p = &years;
```

1245000:	20	years
1245004:		
1245008:		
1245012:		
1245016:		
1245020:	1245000	p
1245024:		
1245028:		

<u>name</u>	<u>value</u>
years	20
&years	1245000
p	1245000
&p	1245020
*p	20

## زبان برنامه نویسی C (21814)

### Change 'years' using the pointer

```
int years, *p;  
years = 20;  
p = &years;  
*p = 10;
```

1245000:	10	years
1245004:		
1245008:		
1245012:		
1245016:		
1245020:	1245000	p
1245024:		
1245028:		

## زبان برنامه نویسی C (21814)

**Not recommended for beginners!**

```
int years, *p;  
years = 20;  
p = &years;  
*p = 10;  
++*p;
```

because  
equivalent to  
++(\*p);

1245000:	11	years
1245004:		
1245008:		
1245012:		
1245016:		
1245020:	1245000	p
1245024:		
1245028:		

زبان برنامه نویسی C (21814)

## Call by Reference

- Give a function the address of a variable instead of its value.
- Used to:
  1. Change values of variables in the caller's function.
  2. Return one or more values to the caller.

زبان برنامه نویسی C (21814)

## Call by Reference (change values)

```
#include <stdio.h>
void swapints(int *p, int *q); // note declaration

int main( void) {
    int a=2, b=5;
    printf("%d %d\n", a, b);    // prints 2 5
    swapints(&a, &b);           // note actual parameters *
    printf("%d %d\n", a, b);    // now prints 5 2
    return 0;
}

void swapints( int *p, int *q) {
    int tmp;
    tmp = *p;    // save a copy of what was in *p
    *p = *q;     // now *p holds what *q holds
    *q = tmp;    // swap is complete
}
```

C(21814) زبان برنامه نویسی

## On Entering swapints

```
void swapints( int *p, int *q) {  
    int tmp;  
    tmp = *p;  
    *p = *q;  
    *q = tmp;  
}
```

1000:	2	a
1004:	5	b
1008:		
1016:		
1020:		
1024:		
1028:	1000	p
1032:	1004	q
1036:		tmp
1040:		
1044:		
1048:		

زبان برنامه نویسی C (21814)

## First Statement

```
void swapints( int *p, int *q) {  
    int tmp;  
    tmp = *p;  
    *p = *q;  
    *q = tmp;  
}
```

1000:	2	a
1004:	5	b
1008:		
1016:		
1020:		
1024:		
1028:	1000	p
1032:	1004	q
1036:	2	tmp
1040:		
1044:		
1048:		

زبان برنامه نویسی C (21814)

## Second Statement

```
void swapints( int *p, int *q) {  
    int tmp;  
    tmp = *p;  
    *p = *q;  
    *q = tmp;  
}
```

1000:	5	a
1004:	5	b
1008:		
1016:		
1020:		
1024:		
1028:	1000	p
1032:	1004	q
1036:	2	tmp
1040:		
1044:		
1048:		



## Last Statement

```
void swapints( int *p, int *q) {  
    int tmp;  
    tmp = *p;  
    *p = *q;  
    *q = tmp;  
}
```

1000:	5	a
1004:	2	b
1008:		
1016:		
1020:		
1024:		
1028:	1000	p
1032:	1004	q
1036:	2	tmp
1040:		
1044:		
1048:		

زبان برنامه نویسی C (21814)

## Pointers and Arrays

- The inventors of C did a very clever thing.
- **The name of an array is a constant pointer to the first element.**
- Given `int odds[5] = {3,5,7,9,11};`
- We can use the name *odds* as a constant pointer to the beginning of the array.
- Let's look at some of the possibilities.

زبان برنامه نویسی C (21814)

## Memory picture of an Array

```
int odds[] = {3,5,7,9,11};
```

Thus *odds* has value  
1000

1000:	3	odds
1004:	5	
1008:	7	
1016:	9	
1020:	11	
1024:		
1028:		
1032:		
1036:		
1040:		
1044:		
1048:		

زبان برنامه نویسی C (21814)

## Pointer Heaven (Hell)

```
int odds[ ] = {3,5,7,9,11}, *p, n;  
p = odds;           // p points to 3  
p = &odds[0];       // same thing  
n = *p;             // n is now 3  
n = p[0];           // same thing—array notation  
                    //on pointers (to arrays)  
p[4] = 13;          // odds[4] is now 13
```

زبان برنامه نویسی C (21814)

## Pointer Arithmetic Example

```
int odds[5] = {3,5,7,9,11}, *p, n;  
p = odds;           // p points to 3  
++p;                // p points to 5  
p = (odds + 1);      // p points to 5  
p = (odds + 4);      // p points to 11  
p = &odds[4];        // same thing, points to 11
```

زبان برنامه نویسی C (21814)

## Pointer Arithmetic Rule

- The unit change in a pointer in arithmetic operations is equal to the size of the variable type it points to.

pointers to *int* change in units of *sizeof(int)*

pointers to *float* change in units of *sizeof(float)*

pointers to *double* change in units of  
*sizeof(double)*

pointers to *char* change in units of *sizeof(char)*

زبان برنامه نویسی C (21814)

## Array Parameters for Functions

- In prototype or definition, type required.
- In call to function, only name of array.
- Given declaration *int array[5];*

**float** ave(**float** array[], **int** n) // **prototype/def**

**float** ave(**float** \*array, **int** n) // **same thing**

**x = ave(array, n);** // **call**

زبان برنامه نویسی C (21814)

## Call by Reference (Return Mult. Values)

```
void minMax(int a[], int n, int *min, int *max) {  
    int i;  
    // initialize to a value in the array  
    *min= *max = a[0];  
    for(i=0; i < n; i++) {  
        if ( a[i] > *max) //if larger, replace with new value  
            *max = a[i];  
        if(a[i] < *min) //if smaller, replace with new value  
            *min = a[i];  
    }  
    return;  
}
```



زبان برنامه نویسی C (21814)

## Icons and Functions

- **2-D array passing by reference.**
- **Read icon data array from a file.**
- **Use a function to draw an icon.**

زبان برنامه نویسی C (21814)

## IconFunctions - readData

```
void readIcon(FILE *fp, char icon[][ICONSIZE]) {  
    int row, col;  
        // loop through all rows and columns  
    for(row = 0; row < ICONSIZE; row++)  
        for(col = 0; col < ICONSIZE; col++)  
            icon[row][col] = fgetc(fp); // store char in icon  
    fgetc(fp); // discard the newline at the end  
}
```

زبان برنامه نویسی C (21814)

## IconFunctions - drawIcon

```
void drawIcon (char icon[][ICONSIZE], int x, int y) {  
    int row, col;  
    // draw each pixel of the icon  
    for (row = 0; row < ICONSIZE; row++)  
        for (col = 0; col < ICONSIZE; col++) {  
            _gotoxy(x + col, y + row);  
            printf("%c", icon[row][col]);  
        }  
}
```

C(21814) زبان برنامه نویسی

## IconFunctions main

```
int main( void ) {  
    char icon[ICONSIZE][ICONSIZE];  
    FILE *fp; int i;  
        // open the data file  
    if ((fp = fopen("icondata.txt", "r")) == NULL){  
        printf("Can't open the icon file");  
        exit (1); // if can't open, have to quit  
    }  
    readIcon(fp, icon); // read pixels into icon array  
    for (i=0; i < 10; ++i) {  
        drawIcon(icon, 6*i, 2*i); // draw icon in diag. row  
    }  
    fclose(fp);  
    return 0;  
}
```



بِنام خدا

زبان برنامه نویسی C (21814)

**Lecture 12:**  
**Selected Topics (Part II)**

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814)

مثال هایی

در مورد کاربرد های Pointers

## زبان برنامه نویسی C (21814) Dynamic Memory Allocation

### تخصیص حافظه بصورت دینامیک:

- هدف گرفتن حافظه از سیستم عامل است.
- مساله: در زبان برنامه نویسی C، مطابق آنچه تا کنون بیان شد، طول آرایه ها در ابتدای برنامه تعیین می شود. اگر بتوان به روشی طول آرایه ها را هنگام اجرای برنامه، نه هنگام کامپایل کردن آن تعیین کرد می توان از حافظه استفاده بهتری نمود.
- این روش در سیستم عامل Windows با گرفتن حافظه برای آرایه انجام می شود.

## زبان برنامه نویسی C (21814) Dynamic Memory Allocation

- هر برنامه ای در **Windows98** یا **Windows XP** می تواند تا **4 GBytes** از حافظه را استفاده کند.
- لذا بایستی در داخل برنامه امکاناتی را فراهم نمود تا از این امکانات سیستم عامل استفاده شود.
- فرایند '**memory allocation**' در مورد سیستم عامل برای این منظور کاربرد دارد.



## زبان برنامه نویسی C (21814) Dynamic Memory Allocation

- در این راستا تابعی بصورت

**void \* malloc ( int size)**

- وجود دارد که این کار را انجام می دهد.
- در این جا **size** مقدار فضای مورد نیاز می باشد.
- این تابع یک **pointer** که به اولین **byte** حافظه تخصیص یافته جدید اشاره می کند را باز می گرداند.
- نوع این **pointer** می تواند هر نوعی باشد ( **char \***, **int \***, **etc.**)
- حافظه تخصیص یافته می تواند به هر نحوی به کار رود.

## زبان برنامه نویسی C (21814) Dynamic Memory Allocation

مثال:

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

#define SIZE 1000
int main(void) {

    int *p, i;
    p = (int *)malloc(SIZE * sizeof(int));
    // p can be considered an array of SIZE ints
    for (i=0; i < SIZE; ++i)
        p[i] = rand(); // init to random numbers
    free(p);
    return 0;
}
```

## زبان برنامه نویسی C (21814) Dynamic Memory Allocation

- با استفاده از **void free(void \* ptr)** می توان حافظه تخصیص یافته را مجددا پس از استفاده آزاد کرد.
- **ptr** در واقع **pointer** است که به ابتدای حافظه تخصیص یافته اشاره می کند.
- برنامه هایی که حافظه گرفته شده را آزاد نکنند، اصطلاحاً برنامه های دارای نشتی حافظه (**memory leak**) نام دارند.
- حافظه های آزاد نشده غیر قابل استفاده بوده و اصطلاحاً **garbage** نامیده می شوند.

## زبان برنامه نویسی C (21814)

قسمتی از یک برنامه را برای دو سوال زیر در نظر

بگیرید: `char a[ ] = "abc", *p = a;`

مقادیر `p[2]` و `*p` بترتیب کدامند؟

- |                         |                       |      |
|-------------------------|-----------------------|------|
| <code>p[2] = 'b'</code> | <code>*p = 'a'</code> | الف- |
| <code>p[2] = 'c'</code> | <code>*p = 'a'</code> | ب-   |
| <code>p[2] = 'b'</code> | <code>*p = 'b'</code> | ج-   |
| <code>p[2] = 'c'</code> | <code>*p = 'b'</code> | د-   |

## زبان برنامه نویسی C (21814)

قسمتی از یک برنامه را برای دو سوال زیر در نظر

بگیرید: `char a[ ] = "abc", *p = a;`

مقادیر `++(*p)` و `*(++p)` بترتیب کدامند؟

- |      |                           |                           |
|------|---------------------------|---------------------------|
| الف- | <code>++(*p) = 'b'</code> | <code>*(++p) = 'b'</code> |
| ب-   | <code>++(*p) = 'a'</code> | <code>*(++p) = 'b'</code> |
| ج-   | <code>++(*p) = 'b'</code> | <code>*(++p) = 'a'</code> |
| د-   | <code>++(*p) = 'a'</code> | <code>*(++p) = 'a'</code> |

## زبان برنامه نویسی C (21814)

- هنگام تعریف متغیر های `pointer` لازم نیست که علامت `*` به نام متغیر بچسبد. در صورت وجود `space` همچنان متغیر بصورت `pointer` شناخته می شود.
- هنگام برنامه نویسی بهتر است برای سهولت درک برنامه، متغیر های `pointer` با اضافه کردن `ptr` به آخر نام متغیر از سایر متغیر ها متمایز شوند.



بِنامِ خدا

زبان برنامه نویسی C (21814)

## Lecture 13

### Chapter 13: Strings

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814) فصل سیزدهم

- در این فصل نحوه ترکیب data type های معرفی شده در فصول قبلی با هم به منظور ایجاد برنامه های پیچیده تر مورد بررسی قرار می گیرند.



# Strings (and Things)

## زبان برنامه نویسی C (21814) فصل سیزدهم

### Arrays of characters

- آرایه ای شامل کاراکتر را بصورت زیر در نظر بگیرید:

```
char c[11] = {'c','h','a','r','s','t','r','i','n','g','s'};
```

c	c
	h
	a
	r
	s
	t
	r
	i
	n
	g
	s

# زبان برنامه نویسی C (21814) فصل سیزدهم

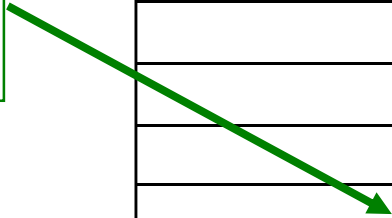
## Strings

- خروجی عبارت زیر را در نظر بگیرید:

```
printf("%s", "charstrings");
```

c
h
a
r
s
t
r
i
n
g
s
\0

'\0' has numerical value 0.  
It's called the **null character**.



زبان برنامه نویسی C (21814) فصل سیزدهم

## Null Pointer vs. Character

- Don't confuse these two

**NULL** = null pointer = address (4 bytes) with value zero. It's a constant defined in `<stdio.h>`. Returned by *fopen()* when error.

**'\0'** = null character = character (1 byte) with value zero. Used as last character of a string to indicate it's end.

## زبان برنامه نویسی C (21814) فصل سیزدهم

### Strings have null termination

- The character '\0' (null character) terminates every string (ASCII value 0).
- The purpose is to delimit (i.e. mark) the end of the string.
- This allows functions (like printf()) to process strings without knowing length ahead of time.

زبان برنامه نویسی C (21814) فصل سیزدهم

## Strings = Arrays of chars

- **A string is a null character terminated array of characters.**
- The *value* of a string is the address of it's first character!
- Thus a constant string functions as a constant pointer.

## زبان برنامه نویسی C (21814) فصل سیزدهم

# Strings and Pointers

- **Initialization in declaration**

```
char a[] = "xyz";
```

```
char a[] = {'x', 'y', 'z', '\0'}; // same thing
```

```
char *p = "xyz"; // pointer initialization
```

- **Assignment**

```
char *p;
```

```
p = "xyz"; // p points to 'x'
```

## زبان برنامه نویسی C (21814) فصل سیزدهم

### No automatic copy in C

```
char a[50], *p, b[ ] = "abcdef";
```

```
p = b;      // OK, assignment of address
```

```
p = "xyz";  // OK, addr. assignment again
```

```
a[ ] = b[ ]; // NO! doesn't work
```

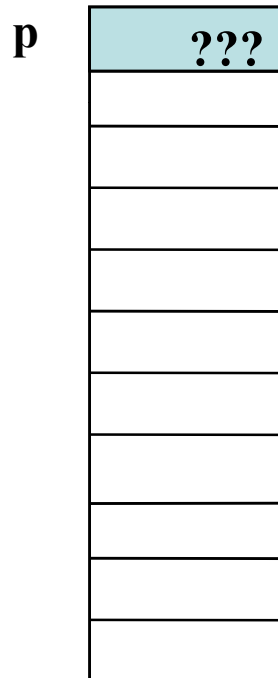
```
a = b;      // NO! a is a constant
```



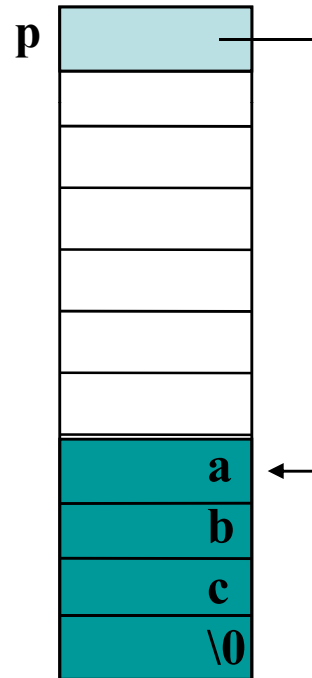
# زبان برنامه نویسی C (21814) فصل سیزدهم

## string declaration and memory

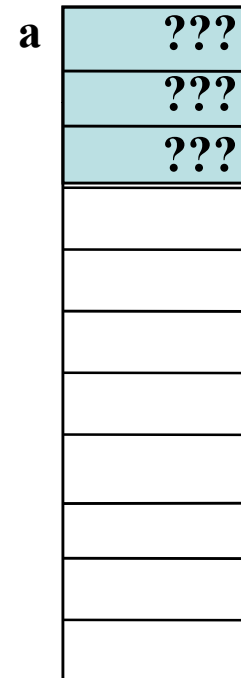
**char \*p;**



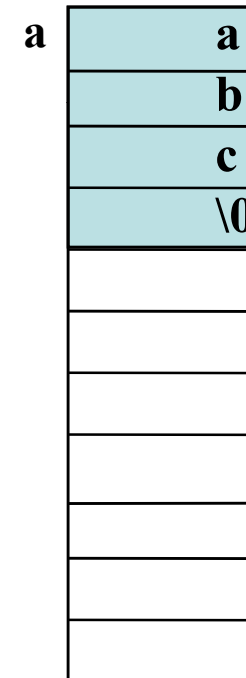
**char \*p = "abc";**



**char a[3];**



**char a[] = "abc";**



## زبان برنامه نویسی C (21814) فصل سیزدهم

### Using a pointer to process a string

- Finding the length of a string

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

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

```
int len(char *p) {  
    int count=0;  
    while (*p++) ++count; /* counts chars in string */  
    return count;  
}
```

null char at end of string  
makes the loop stop.

```
void main(void){  
    char *s = "12345";  
    printf("%d", len(s)); /* prints number 5 */  
    printf("\n\n");  
}
```

## زبان برنامه نویسی C (21814) فصل سیزدهم

### String handling functions

- **scanf(), gets(), fgets():** in **<stdio.h>**.

These require **#include <string.h>**

- **strlen():** length of a string.
- **strcpy():** copy a string.
- **strcat():** concatenate a string.
- **strchr(), strrchr():** search for a character.
- **strstr():** search for a substring.
- **strcmp():** compare two strings.

## زبان برنامه نویسی C (21814) فصل سیزدهم

**int scanf(const char \*, ...)**

```
char a[80];  
scanf("%s", a); // could use scanf("%s", &a[0])  
for( int i=0; i<=80;i++){  
    printf("%c" ,a[i]);  
}
```

- Reads from KB until **white space encountered**. Copies into a[ ].
- **"This is a string"** as input will fill a[ ] with **"This"** only.
- Thus can be only used to input single words.

## زبان برنامه نویسی C (21814) فصل سیزدهم

**char \*gets(char \*)**

**char a[80];**

**gets(a);**

- **Copies characters from keyboard into a[ ] up to but not including the newline.**
- **Copies everything including white space.**
- **Puts '\0' at end.**
- **Must be enough room in a[ ] for the input!**

## زبان برنامه نویسی C (21814) فصل سیزدهم

**char \* fgets(char \* str, int n, FILE \* fp)**

- Reads a string from a file and stores in array pointed to by *str*.
- Stops reading when *n* characters read or when '\n' read, whichever comes first.
- The '\n' will be stored in the string.

زبان برنامه نویسی C (21814) فصل سیزدهم

**int strlen(const char \* s)**

**char \*s = "This string is 27 char long"**

**printf("%d", strlen(s));**

- **Prints the value 27**
- **Does not count the '\0' at the end.**

زبان برنامه نویسی C (21814) فصل سیزدهم

**char \*strcpy(char \* s, const char \* t)**

**char s[10], t[ ] = "12345";**

**strcpy(s, t);**

**printf("%s", s); /\*or printf("%s", strcpy(s,t)); \*/**

- prints the string "12345"
- You must make sure s[ ] is big enough to hold all of the characters of t[ ]!
- if declared s[3], then **BIG TROUBLE**.



زبان برنامه نویسی C (21814) فصل سیزدهم

**int strcmp(const char \* s, const char \* t)**

**char s[ ]="smith", t[ ]="smyth";**

**n = strcmp(s, t);**                      **/\* n is negative \*/**

**n= strcmp(t, s);**                      **/\* n is positive \*/**

**n = strcmp(s, s);**                      **/\* n is zero, \*/**

- **This function is used for sorting words alphabetically.**

## زبان برنامه نویسی C (21814) فصل سیزدهم

**char \*strcat(char \*s, const char \* t);**

**char s[20], t[] = "bea", u[] = "vers";**

**strcpy(s, t);**      **/\* copies t to s \*/**

**strcat(s, u);**      **/\* s[] now contains "beavers" \*/**

- Appends (concatenates) second string onto end of first (starting at '\0' position).
- You must have room in s[] or again there will be **TROUBLE**.
- If s[3] declared, **it won't work (sometimes)**.

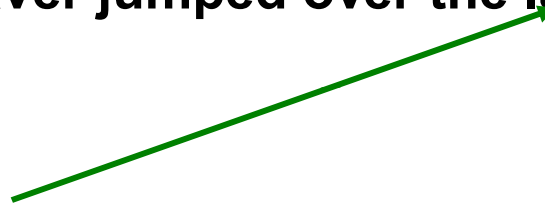
**char \*strchr(const char \* s, int ch)**  
**char \* strrchr(const char \* s, int ch)**

- Searches s for a character and returns pointer to the character -- strchr() from the start, strrchr() from the end.

char s[] = "The quick sly beaver jumped over the lazy white duck";

strchr(s, 'y') points to here.

strrchr(s, 'y') points to here.



زبان برنامه نویسی C (21814) فصل سیزدهم

**char \*strstr(const char \* s, const char \* t)**

- Searches s for a substring, t and returns pointer to the beginning of the substring

**char s[ ] = "The quick sly beaver jumped over  
the lazy white duck";**

**strstr(s, "jumped") points to here.**



زبان برنامه نویسی C (21814) فصل سیزدهم

## Monkey Works

- **Given enough time could a monkey typing randomly, create Hamlet?**
- **See how long it takes a monkey to type a given word.**
- **Make a function that types randomly until it matches a given word.**

## زبان برنامه نویسی C (21814) فصل سیزدهم

### Monkey Works

```
int main(void) {  
    char name[20];  
  
    printf("type a name: ");           // prompt the user  
    scanf("%s", name);                // for a name  
  
    // and print out the results  
    printf("\n\nIt only took him %d tries to type %s",  
        fastMonkey(name), name);  
  
    return 0;  
}
```

## زبان برنامه نویسی C (21814) فصل سیزدهم

### fastMonkey function

```
int fastMonkey(char *name) {  
    char alpha[] = "abcdefghijklmnopqrstuvwxyz";  
    char test[20]; // the output of the random typing  
    int done = 0, count = 0, i;  
  
    while(!done) { // loop until typing is successful  
        for(i = 0; i < strlen(name); ++i) // create test same length as  
            'name'  
            test[i] = alpha[rand() % 26];  
        test[i] = '\0'; // make 'test' a string  
        if(strcmp(test, name) == 0) // check if matches  
            done = 1;  
        ++count;  
        if(count % INC == 1) { // print out progress  
            printf("%d ", count/INC);  
            fflush(stdout);  
        }  
    }  
    return count;  
}
```

زبان برنامه نویسی C (21814) فصل سیزدهم

## myScanf for Strings

- Make a function that acts like *scanf("%s", word)* given the declaration.  
**char word[30];**
- i.e. reads word from KB and puts in *word[ ]*



## زبان برنامه نویسی C (21814) فصل سیزدهم

### myScanf() code

```
int myScanf(char *s) {  
    int inChar;  
  
    while (isspace(inChar = getchar()))  
        ; // skip white space  
    while ( !isspace(inChar) && inChar != EOF) {  
        *s++ = inChar; // copy while not whitespace or EOF  
        inChar = getchar(); // and get the next char  
    }  
    *s = '\0'; // make it a string  
    if( inChar == EOF)  
        return EOF; // if at end, tell the caller  
    else  
        return 1; // number of words found  
}
```

زبان برنامه نویسی C (21814) فصل سیزدهم

## Use myScanf to Count Words

```
int main (void)
{
    char word[30]; int count = 0;

    while(myScanf(word) != EOF){
        ++count;    // get words from KB and count
                   // printf("%s\n", word);
    }
    printf("\nCounted %d words\n\n", count);
    return 0;
}
```

## زبان برنامه نویسی C (21814) فصل سیزدهم

### EC6 -- myScanf() for Integers

- make a *myScanf()* function that works like *scanf("%d", &val)* for positive integers.
- i.e. *myScanf(int \* p)* reads digit characters from KB (use *getchar()*) and converts to integer.
  - skips white space.
  - reads digits and converts to integer until
  - non-digit read, then stores in integer at \*p
- You should be able to replace a call to *scanf("%d", &val)* with *myScanf(&val)*.

بِنامِ خدا



دانشگاه صنعتی شریف

زبان برنامه نویسی C (21814)

**Lecture 14**

**Structures**

مدرس: ناصر سلماسی

## زبان برنامه نویسی C (21814)

# Structures

- یکی از پرکاربردترین روشهای ترکیب کردن داده ها در زبان C
- برخلاف حالت استفاده از یک آرایه، می توان داده ها (متغیرها) با انواع مختلف را در یک جا استفاده نمود.
- هر یک از بخش های یک Structure برخلاف آرایه ها دارای نام جداگانه است.

## زبان برنامه نویسی C (21814) - Structure

- از کاربرد های مهم این روش استفاده از آن برای ایجاد بانک های اطلاعاتی است.

- در این جلسه

- نحوه نمایش Structure

- عملیات های قابل اجرا به روی آنها

- نحوه ترکیب آرایه ها با آنها

- مورد بررسی قرار می گیرند.

## زبان برنامه نویسی C (21814) - Structure

- استفاده از Structure ها این امکان را می دهد که خاصیت های مختلف یک شی/فرد/ مورد مطالعه به صورت مرتبط با هم ارائه شوند.

- مثال: اطلاعات مختلف یک ورزشکار در بانک اطلاعاتی فدراسیون های ورزشی.

- یک متغیر Structure شامل تعداد متعددی فضای تعریف شده (slots) است که اعضای Structure نامیده می شوند (members or fields).

## زبان برنامه نویسی C (21814) - Structure

- هر یک از اعضای Structure بیانگر یکی از خصوصیات شی مورد نظر است.
- در زبان های به کار رفته در بانک های اطلاعاتی، یک Structure به نام یک Record نامیده می شود.



## زبان برنامه نویسی C (21814) - Structure

### Structure Declaration

```
struct tag {  
    type variable;  
    type variable;  
    ...  
} variable, variable, ...;
```

The diagram illustrates the syntax of a structure declaration in C. It shows the following components and their annotations:

- called members**: An orange box with two orange arrows pointing to the *type variable* declarations inside the curly braces.
- optional (but need at least one)**: A red box with a blue arrow pointing to the *...* line, indicating that at least one member is required.
- variable list**: A green bracket under the *variable, variable, ...* part of the closing brace, indicating the list of variables of the structure type.

## زبان برنامه نویسی C (21814) - Structure

- یک **Structure type** شامل
  - **Structure type specification** با کلمه **struct**.
  - یک نام اختیاری که **tag name** نامیده می شود.
  - تعریف اعضا در داخل **{ }**.
  - نام متغیر ها (اختیاری).

## زبان برنامه نویسی C (21814) - Structure

- نحوه تعریف اعضا دقیقا همانند روش تعریف یک متغیر است.
- دو عضو یک **structure** می توانند دارای **type** مشابه باشند.
- هر یک از اعضای یک **structure** می توانند بصورت مجزا صدا زده شوند.
- هر یک از اعضا بایستی دارای نام منحصر به فردی باشند.

## زبان برنامه نویسی C (21814) - Structure

مثال:

```
struct course {  
    char * dept;    /* e.g. "IE" */  
    char * title;   /* e.g. "Intro to C Programming" */  
    int number;     /* e.g. 21814 */  
    int section;    /* e.g. 1 */  
};
```

- This defines a type only (struct course), no memory used or variables defined. **This is not a declaration!!**

## زبان برنامه نویسی C (21814) - Structure

### Variables of type 'course'

```
struct course {  
    char * dept;    /* e.g. IE */  
    char * title;   /* e.g. Intro to C Programming*/  
    int number;     /* e.g. 21814 */  
    int section;    /* e.g. 1 */  
} c1, c2, c3;
```

- Defines *and* Declares three 'course-type' variables.

## زبان برنامه نویسی C (21814) - Structure

### نحوه مقدار دهی به اعضای Structure:

```
c1.dept = "IE"; /* called 'dot' notation */  
c1.title = "Intro to C Programming";  
c1.number = 21814;  
c1.section = 1;  
printf("%s %s %d %d\n", c1.dept, c1.title,  
      c1.number, c1.section);
```

**IE Intro to C Programming 21814 1**

## زبان برنامه نویسی C (21814) - Structure

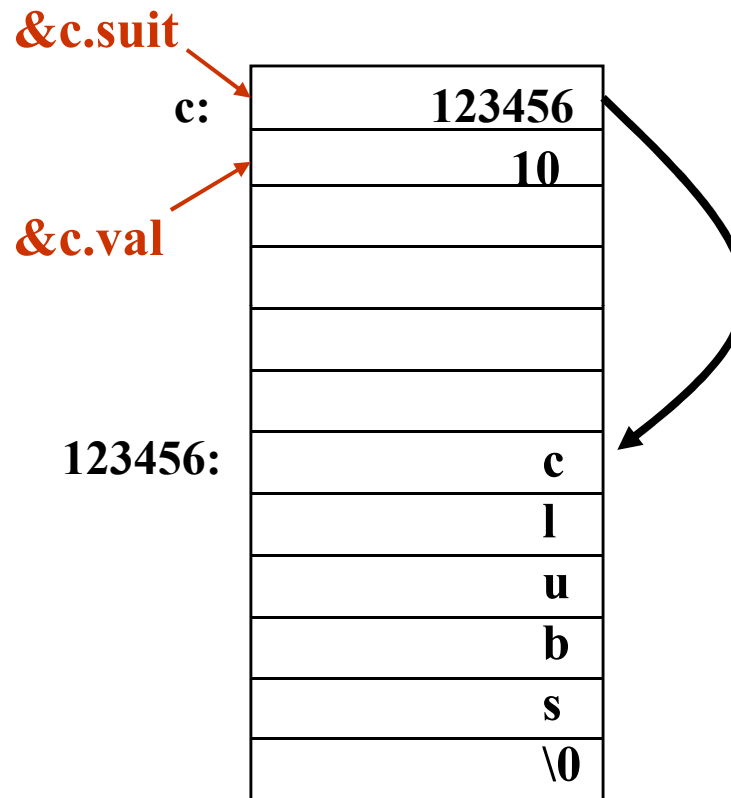
### تفاوت میان نوع Structure و نحوه تعریف آنها

```
struct course {          /* this defines the new 'struct course' type */
    char * dept;          /* e.g. IE */
    char * title;         /* e.g. Intro to C Programming*/
    int number;           /* e.g. 21814 */
    int section;          /* e.g. 1 */
};
```

```
struct course courses[100]; /* an array (collection) of 100 courses */
courses[0].dept = "English"; /* note syntax */
courses[1].dept = "Psychology"; /* etc. */
```

## زبان برنامه نویسی C (21814) - Structure

```
struct card {  
    char *suit;  
    int val;  
} c;  
  
c.suit = "clubs";  
c.val = 10;
```





## زبان برنامه نویسی C (21814) - Structure

```
struct card {  
    char *suit;  
    int val;  
} c;
```

این دو روش مانند هم می باشند:

```
struct card {  
    char *suit;  
    int val;  
};  
struct card c;
```

## زبان برنامه نویسی C (21814) - Structure

- هنگام تعریف یک Structure می توان در حالت هایی مانند زیر نام Structure را تعریف نکرد.

```
struct {  
    char * name;  
    char *tel_no;  
} tel_list[100]; /* list of 100 names & telnos */
```

- در چنین حالتی فقط یک نوع متغیر به این صورت تعریف شده است.

## زبان برنامه نویسی C (21814) - Structure

- هنگام تعریف یک Structure، نوشتن حداقل یکی از این دو ضروری است.

```
struct {  
    char * name;  
    char *tel_no;  
};
```

- **This is useless** -- no variables of this type and no tag to use to declare variables.

## زبان برنامه نویسی C (21814) - Structure

### typedef declaration

- یکی از روشهای مناسب برای نام گذاری **type** های موجود استفاده از **typedef** است.

**typedef** *old\_type new\_type\_name;*

- Can be used to rename all current types.
- **char**, **int**, **float**, **double**, **struct**, and pointers to these.
- Mostly used with **struct** type

## زبان برنامه نویسی C (21814) - Structure

```
typedef struct {  
    char * dept;  
    char * title;  
    int number;  
    int section;  
} course_t;  
  
/* course_t is new type and can be used in declarations */  
  
int main(void) {  
    course_t IE21814, cs161, cs162; /* declares 3 courses */  
    IE21814.title = "Introduction to C Programming";  
    etc.
```

## زبان برنامه نویسی C (21814) - Structure

نحوه مقدار دهی به اعضای Structure:

مثال:

```
typedef struct Lumber{  
    char type[11];    //type of wood  
    int height, width, length;  
    float price;  
    int quantity;  
} lumber_t;  
lumber_t sale;  
lumber_t plank = {"white oak", 1, 6, 8, 5.82, 158};
```

## زبان برنامه نویسی C (21814) - Structure

### Typedef or Tag?

```
typedef struct {  
    char * dept;  
    char * title;  
    int number;  
    int section;  
} course_t;
```

```
/* 'course_t' is type*/  
course_t c1, c2, c3;
```

```
struct course {  
    char * dept;  
    char * title;  
    int number;  
    int section;  
};
```

```
/* 'struct course' is type*/  
struct course c1, c2, c3;
```

## زبان برنامه نویسی C (21814) - Structure

### توجه:

**در زبان برنامه نویسی C دستوری برای مقایسه دو Structure وجود ندارد. در صورت نیاز به مقایسه دو structure بایستی برنامه نویسی انجام داد.**



## زبان برنامه نویسی C (21814) - Structure

```
typedef struct {  
    char * dept; char * title;  
    int number; int section;  
} course_t;  
  
int main(void ) {  
    course_t c, *p;  
    p = &c;      /* dept member in struct pointed to by p */  
    p->dept = "cs";  
    p->number = 151;  
    printf("%s %d", c.dept, c.number);  
    return 0;  
}
```

prints: cs 151
-------------------

## زبان برنامه نویسی C (21814) - Structure

مثال:

```
typedef struct Lumber{  
    char type[11]; int height, width, length;  
    float price;    int quantity;  
} lumber_t;  
lumber_t plank = {"white oak", 1, 6, 8, 5.82, 158};  
  
float cost;  
Lumber_t *p;  
P = &plank;  
Cost = p->price;
```

## زبان برنامه نویسی C (21814) - Structure

روشهای دستیابی به اعضای یک **structure**:

— دسترسی مستقیم

```
printf(“ %i \n “,plank.height);
```

— استفاده از **pointer**: در این روش از **arrow operator** برای دسترسی به اعضای **structure** استفاده می شود.

```
printf( “ %i \n”, p-> height);
```

## زبان برنامه نویسی C (21814) - Structure

روش انتقال مقادیر یک **structure** به دیگری:

```
typedef struct Lumber{  
    char type[11];  int height, width, length;  
    float price;    int quantity;  
} lumber_t;  
lumber_t sale;  
lumber_t plank = {"white oak", 1, 6, 8, 5.82, 158};  
  
sale = plank;
```

## زبان برنامه نویسی C (21814) - Structure

### نحوه انتقال structure ها به function ها:

- می توان structure ها و pointer هایی که به یک structure اشاره می کنند را به عنوان argument های یک function به کار برد.
- از آنجا که یک structure دارای اجزای متعددی می تواند باشد، بهتر است که pointer آن استفاده شود.

## زبان برنامه نویسی C (21814) - Structure

```
typedef struct { /* note: comes before print_card prototype */
    char * suit;
    int val;
} card_t; /* function prototype uses 'card' type */

void print_card(card_t c);

int main(void) {
    card_t c = {"clubs", 10}; /* note: initialization! */
    print_card(c); /* call by value */
    return 0;
}

void print_card(card_t c) { /* this c is a copy of the c in main */
    printf("%d of %s\n", c.val, c.suit);
}
```

## زبان برنامه نویسی C (21814) - Structure

```
typedef struct { char * suit;  int val; } card_t;

void print_card(card_t c);
void init_card(card_t *p, char * suit, int val);

int main(void) {
    card_t c;
    init_card(&c, "clubs", 10); /* call by reference */
    print_card(c);             /* call by value */
    return 0;
}

void print_card(card_t c) {
    printf("%d of %s\n", c.val, c.suit);
}

void init_card(card_t *p, char * suit, int val){ /* p points to c in
    main */
    p->suit = suit;
    p->val = val;
}
```

## زبان برنامه نویسی C (21814) - Structure

### تمرین:

برنامه ای بنویسید که لیست دانشجویان، به همراه اطلاعات مربوطه را از یک فایل ورودی گرفته، میانگین و واریانس امتحان های میان ترم و پایان ترم را محاسبه کند. سپس از کاربر نحوه مرتب کردن لیست را (برحسب نام، نام خانوادگی، شماره دانشجویی، نمره میان ترم، و یا نمره پایان ترم) مرتب نماید. در این برنامه اطلاعات دانشجویان را در یک **structure** ذخیره کنید.



## زبان برنامه نویسی C (21814)

### تمرین:

مناسب ترین روش صدا زدن یک تابع بصورت

prototype ***void myFun( student\_t x);***

کدام است؟

- الف- myFun (Ali\_Jamshidi)
- ب- myFun (&Ali\_Jamshidi)
- ج- myFun (student\_t Ali\_Jamshidi)
- د- myFun (\*Ali\_Jamshidi)