دانشگاه آزاد اسلامی واحدابهر

جزوه برنامه نویسی به زبان C

استاد:

آقای دکتر رشتچی

تهیه و تنظیم:

مرتضی کلانتری آزاد

بهار85

مقدمات برنامه نویسی به زبان C

مجموعه کاراکترها : - کارکاترهای A,Z , Q , S ارقام 9-o 1100 برنامه نویسی به زبان C

- مجموعه سمبلهای خاص ! * + / " <
- بعضی از کامپایلی ها $ و @ > } =) % ٪ (~ ؛ { / ^ - [: ، ؟ & -] ، Blank.

شناسه ها و کلمات کلیدی identifiers and key words

شناسه اسمی است که به یک المان برنامه چون متغیر ، اسم تابع ، رشته و..... داده می شود مجموعته ای از حروف و ارقام و .... است که حرف اول عدد نمی تواند باشد.

C بین حروف کوچک و بزرگ تمایز قائل است.

x yg Qvevag. 4^{th}

شناسه های غلط cross – name شناسه های معتبر Stydent-1

Uy دو شناسه کاملا متفاوت NAME ، name

حداکثر طول هر شناسه می تواند 3_1 کاراکتر باشد ، در انتخاب اسامی دقت شود اسامی با مفهوم و مضرانه مختصر و نه خیلی بلند انتخاب شوند.

Name → n

1- خوانائی برنامه

حسن انتخاب 2- قابل استفاده بودن مناسب برای دیگران student name →st-name

مناسب اسامی 3- غلط یابی و تصیح و تغییر راحت تر طولانی teacher name →te- name

Key word: شناسه ئی می باشند که در زبان c (کامپایلی) مورد استفاده قرا گرفته و مجاز به استفاده از آنها نمی باشیم.

C استاندارد دارای 32 Key word بوده و بعضی از کامپایلی ها کلمات کلیدی دیگری را نیز اضافه کرده اند مثل:

Asm	short
Break	white
Case	for
Char	: asm
Int	
Float	
:	

انواع داده ها در زبان C

- int صحیح
- Char کاراکتری

Float اعشاری

Double اعشاری با دقت مضاعف

Void

Short ترکیب شوند. Rang اعدادی که برای هر نوع بکار می رود بیشتر تابع کامپایلی می باشد.

Nit	unsigned	o, 2 n -1
Signed	2-1 $^{n-1}$	, -2 $^{n-1}$

محدوده مجاز معمول برای هر نوع به شرح زیر است با ز ، قابل قبول

نوع پیش فرض توسط کامپایلی هم مشخص می شود . 255-0 اندازه بیت char

8 - 128 و 127

اگر uns : grad بیان نشود پیش فرض نوع علامت دار است.

Int 16 o, 2^{16-1}

INS: grad int 16 2^{15} -1, - 2^{15}

Short int 16

ins : gred short int 16 مقادیر ثابت عددی

ثابتهای صحیح اگر ثابت صحیح استفاده شده خارج از rang متغییر باشد اشکال ایجاد می شود.

Fong int 32

Flout 32 3.4E -38 3. 4 E -308 o 740 32 70

Double 64 1.7E – 368 1.7E308

long double 128 ϕ ϕ 1 0743

داده های ثابت مبنای 8 با C شروع می شوند.

داده های صحیح مبنای 16 با Qx شروع می شوند.

ثابتهای اعشاری $\phi x1\phi xQ\phi x1a\phi$

دقت در عملیات اعشاری با ؟ G . o. 0001

! از اعداد اعشاری نمی توان به عنوان پارامترهای مقایسه و نتیجه گیری استفاده کرد .

ثابتهای کاراکتری در داخل اسپرتف apostrophes بیان می شوند .

کدهای اسکی Askl

3American standard code formation interchange ≠ 51 65 کد

اسکی کدهای 128 تا 255 استاندارد نیستند.

کد غیر قابل چاپ O HT = 09 cR = 13

` LF = 10 Esez 27

: VT = 11

31 FF – 12

Escape sequence3

برخی از کاراکترهای غیر قابل چاپ و کاراکتر هایی چون " ، , ، ؟ ، / در غالب Escape sequence بیان می شوند .

نحوه ورود کاراکترها با استفاده از کدهای اسکی

و کلید HT

/ a	bell	Q7
/b	backspace	Q8
/ t	H.T	Q9
/ V	V. T	Q11
/ n	new fine	1 Q
/ F	form feed	12
/r	C.R carriage return	13
/ “	“	34
/ ,	,	39
/?	?	63
//	/	22
/Q	hull	QQ

بیان انتهای یک رشته

ثابتهای رشته ای یک ثابت رشته ای مجموعه ای از چندین کاراکتر است که بین دو "

" واقع می شوند .

"ZAND; AN university " Name 1

" Name 1 /n Name 2/n Name 3 " Name 2 هنگام چاپ

Name 3

کامپایلر در انتهای هر کار اکتر رشته ای کاراکتر null یا /Q را با کد QQ بصورت اتوماتیک اضافه می کند .

u	n	I	v	E	R	s	I	T	Y	Null	

85	78	---- --- ---- --- ---- -- --	QQ		

1	0	0	0	1	1	1	0

" A" ≠ ' A '

65 00 65

متغیر ها و بردارها

متغیر شناسه یا اسمی است به نوع خاصی از اطلاعات در بخش خاصی از برنامه داده می شود . اطلاعات با اسم متغیر های خود قابل دسترسی می باشند .

محتوای اطلاعاتی یک متغیر در داخل برنامه و در هنگام اجرا هم می تواند تغییر کند ولی نوع آن قابل تعریف نیست.

بردار ، آرایه array b=1

A = 31

D=” w “

C = a –b

شناسه ای است که به مجموعه ای از اطلاعات داده می شود که همگی دارای یک اسم می باشند همه اطلاعات می بایست از یک نوع باشند .

هر کدام از عناصر بردار با یک زیر نویس اندیس قابل دسترسی است .

اندیس ها در زبان C از ً شروع می شوند . ÷wrcxdnrcxx

تعریف متغیر ها . Dectarations

در زبان c تعریف متغیرها می بایست در ابتدای برنامه و قبل از اولین دستور صورت گیرد.

برای جدا کردن item ها int a , b , c ,

Float roat 1 , roat 2 ,

Char frag, name {40 y; name

پس از تعریف یک متغیر هنگام compile برای هر کدام از متغیرها حافظه مناسب اختصاص داده می شود.

Int a; int a, b;

;Int b Int c;

Int c;

هنگام اجرای برنامه EXE حافظه اشغال می شود.

Char name {500} → compile → EXE

آدرس مربوط به متغیرها را مشخص میکند

بهتر است اسامی بصورت کامل نوشته شوند چون ممکن است بعضی از کامپایلرها اشکال بگیرند. Short int a; short a;

Unsigned long int x;

توجه : ممکن است انواع متغیرهائی که کامپایلر قبول می کند و فضائی که برای هر یک اختصاص می دهد متفاوت باشد. تعریف صحیح نوع متغیرها کاملاً حائز اهمیت است چون فضای حافظه را تلف می کند.

مقدار دهی اولیه متغیرها : هنگام تعریف متغیرها مقدار دهی اولیه را هم می توان انجام داد .

غلط Error یک اسم یا شناسه را فقط برای یک نوع متغیر می توان بکاربرد

```
;Int      c = 12
Char   star = ,*,؛
 Chat c = n;
 Float aver = 1.25 6 – 3
 Name; y = “ALI “
 Name {20} = “ALI”
 Name {3} = “ALI”
 Name {4} = “ALI”
```

یک عبارت می تواند متشکل از یک داده ، عدد یا کاراکتر یا یک موجودیت منمفرد مثل یک متغیر ، آرایه یا اسم یک تابع باشد تا ترکیبی از این موجودیتها توسط یکسری عملکرد ها چه محاسباتی چه شرطی باشند.

A+b	x=y	e=a+b	XL=y	++k	--;
اپراتور جمع	اپراتور انتساب	عملکرد مقایسه ای			

Statementsجملات یک Statements سبب می شود کامپیوتر کاری را انجام دهد .

Expression متشکل از یک Expression و یک ؛ در انتها است.

compound Statements مجموعه ای Statements ها که در داخل { } قرار گرفته اند.

Contran سبب یکسری عملیات کنترلی می شوند.

اجرای یک Statements Expression سبب ارزیابی عبارت می شود.

A=3;

A+I; یک واحد به ؛ اضافه می کند.

X = w+z x w .z {

```
Pi= 3.14;
area = pitradius + radiusi
}
```

```
While (i< = 10)
{
scanf ( "%f" , &X);
sumr = x;
; = ; + 1 ;
}
```

symbolic constant

نامی است که جایگزین مجموعه ای از کاراکترها می شود . کاراکتر ها ممکن است بیانگر یک ثابت عددی ، ثابت کاراکتری یا ثابت رشته ای باشند . در ابتدای برنامه تعریف موی شوند .

define اسم سمبول مقدار

define p I = 3.14

define NAME = "ALI"

توجه : ؛ در انتها ندارد.

برای اینکه بین شناسه های معمولی و اسم سمبلها تمایز وجود داشته باشد اسامی سمبلها با حروف بزرگ نوشته می شوند .

فصل سوم : عملکردهای محاسباتی

٪ خارج قسمت تقسیم صحیح را بیان می کند.

+ addition جمع

– subtraction تفریق

* multiplication ضرب

/ division تقسیم

% remainder after integer division

Q = 13	a +b	17
B=4	a-b	9
A*b	5.2	
A/b	3	
a%b	1	

باید b# 0 باشد.

هر دو اپرند می بایست صحیح باشد و b#0باشد .

X= 14.5	x+y	16.5
Y=2.0	x-y	12.5
2.	X*y	29.0
	X/y	7.25

برای متغیرهای کاراکتری عملکردهای ± می تواند استفاده شود .

C1 =, m,	77	c1	77
C2 =, 1,	49	c1+c2	126
C1 +c2+3			129
C1+c2+, 3,			177

برای عملکرد ٪ اگر یکی از ابرند ها منفی باشد علامت اولین اپرند به باقیمانده نسبت داده می شود.

A=11	a/b→ -3	a = -11	a /b → - 3
B= - 3	a%b 2	b = 3	a % b -2
A= -11	a /b 3		
B= - 3	a % b -2		

اگر دو اپرند از یک نوع نباشند عملیات در بالاترین دقت ممکن صورت نمی گیرد

توصیه نمی شود.

1- float . Doubleاجزای به doubleتبدیل می شوند.

2- char , int , float دو تای آخر به float تبدیل می شوند.

3- int , float آخری به long int تبدیل می شود .

int , I =7 i+f 12.5 float;

f float , f = 5.5 i+c 126 int

c char , c = w i+c –o 78 int

i+c – 2 * f/5 123.8 float

چنین عملیاتی به هیچ وجه توصیه نمی شود .

Type cast برای تبدیل نوع داده data type expression

=7 int (;+f) %4 ;

F= 8.5 float

((Int) (i+f)) % 4

((Int) f) %2

ترتیب اجرای عملیات 1- ابتدا عملیات داخل پارانتزها از داخل ترین پارانتز اجرا می شود .

2- عملیات ٪ و / ، * ترتیب از چپ به راست .

3- عملیات - و + ترتیب از راست به چپ

a- b/c*d = a –{(b/a)+d}

$\frac{a-b}{c*d}$ = (a-b) /((*d))

Unary operator

-750 750 قرینه یا مکمل -

=;+1; = یک واحد اضافه به ; I++ ++increment

=;+1; = یک واحد کاهش از ; I -- decrement - -

++ و – هم قبل از اپرند و هم بعد از اپرند می توانند نوشته شوند .

i=1 printf(“;=%din” , ;) ; ++; k++

i =2 printf(“;=%din” ,++ ;) ; - - ; l --

i =2 printf(“;=%din” ,++ ;) ;

i=1 printf(“;=%din” , ;) ;

i=1 printf(“;=%din” , ;++) ;

i=1 printf(“;=%din” , ;) ;

اپراتور یکانی

size of; i Int ;i

Size of (integer)

Size of (float)

size of text Char tert { } = “university “;

عملکردهای نسبی (مقایسه ای) و منطقی

عملکرد	مفهوم	عملکرد	مفهوم
< =	less them		
< =	less them or equal to		
>	Greater then	= =	equal to
< =	greater then or equal to	b =	not equal to

عملکرد = که برای انتساب بکار می رود با = = کاملاً متفاوت است.

نتیجه عملکردهای فوق به یکی از این دو صورت میباشد. که مقدار برگشتی عددی صحیح است.

True 1

False Q

; = 3 j = 5 k=2

I < j true 1

(I-j) < = k true 1

;! = (j-k) false Q

(; +7) = = (j*k) true 1

عملکردهای منطقی && log: cal and

| | log: cal or

!log : cal not عملکرد یکانی (unary)

(;< j) && (k==2) true ; = 7 c=, w, f=6.5

(;< j) && (j==k) false (;> = 6) && (c= =, w,) true

(j= =5) || (k! ==2) true (;> = 6) || (c= =119) true

! (j<j) false (c! =, p,) && (c>100) true

! (j< j) true! (i> (f-1)) false

j>j true

Identifier = expression

a= 3; x =y; area = length + width;

Identifier 1 = identifier 2 = ---------- = expression

a= b=x=y=3; { a=3; b=3; x=3; y = 3;

فرض کنید ;I . j متغیرهای صحیح با مقادیر j=5 باشد

عبارت	مقدار
; =j	5
; =j/2	2
; =2*j/2	5
; =2*(j/2)	4
; =, x,	120

i=, Q, 48

;=(, x, - , Q,) /3 24

+ =, - = , * = , / = , % =

Expression 2 = expression 2 = expression 2 = expression 2 expression 2

X+ = y x=x+y

Z * = (f-5) z = z*(f-5)

X* = - 2 * (y+ z) / 3 x = x * (-2 * (y+z) /3)

conditional operator عملکرد شرطی

Expression 1 ? Expression 2: expression 3

عبارت اگر دست باشد مقدار عبارت برابر در غیر اینصورت برابر خواهد بود .

(; <Q) ? Q: 100

Mines (f < g) ? f: g ; کوچکترین عدد در min قرار می گیرد.

اولویت اجرای عملکردها

Unary operators	- ++ --! Sizeof (type)	R → L
Arithmetic	* / %	L → R
Arithmetic	+ -	L → R
Velational	< < = > > =	L → R
Equality	= = ! =	L → R
Logical and	&&	L → R
Equality or	||	L → R
Conditional	? :	R → L
Assignment	= + = - = * = / = %=	R → L

( ) دارای بالاترین اولویت می باشد

C + = (a>Q&& Q< = 1Q)? + + a: a / b;

A=m+n x = m+n/ p+(r+s) x= m+ (n/p) * (r+s)

P (k+s)

X= (m+n) /p*(r+s) $x = \frac{(m+n)}{P} * (r+s)$

X= (m+n/ (p*(r+s)) صحیح

توابع کتابخانه : library functions عملکرد

تابع	نوع تابع	
abs (i)	int	قدر مطلق آرگوهان

ceil (d) double روند کردن کوچکترین عدد صحیح را که برابر یا بزرگتر از عدد است

Cos (d) double and Cos (d) double ceahd ; integer

Exp (d) e (e=2.71) d double

fabs (d) double قدر مطلق آرگون اعشاری C character

getchar ( ) int ورود یک کاراکتر از دستگاه ورودی استاندارد

$\sqrt{d}$ Sqrt (d) double

log (d) double lnt لگاریتم طبیعی put char (c) ارسال کاراکتر به دستگاه خروجی

Pow (d1 , d 2) double d 2

Toascii (c) int کد اسکر آرگوهان را حساب می کند.

Tolower (c) int

Tovpper (c) int

استفاده از روش کتابخانه ای و neaer file

Abs (i)	stdlib . h
Cos (d)	math . h
Printg (…)	stdio.h
Clrscr( )	conio . h
Toascii (c)	ctype . h

```
# include <stdio. h >
# main ( ) < canio . h >
{
int lower , upper ;
clrscr ( ) ;
lower = get char ( ) ;
 upper  = topper ( lower )
 put char  ( upper ) ;
 }
```

تمرین : هر یک از دانشجویان حداقل 20 تابع کتابخانه ای را که در کلاس بیان نشده اند را از Help پیدا کرده و عملکرد آنها را بطور مختصر شرح دهند. از هر cheater نباید بیش از 2 تابع مورد بررسی قرار گیرد . سپس سه برنامه به شماره های L3* L2* L1* به اختیار و به گونه ای نوشته شوند که در هر برنامه حداقل سه تابع از

توابع فوق بصورت غیر تکراری مورد استفاده واقع شده باشند . توضیحات 20 تابع می بایست ضمیمه لیست برنامه ها باشد.

تذکر : توضیح داده شود که برنامه ها برای چه منظوری نوشته شده اند.

ورود و خروج اطلاعات

ورود و خروج یک کاراکتر منفرد از stdinput GET CHAR ( ) ;

Char variable = get char ( );

Char c ; cansio , stdinput stdinput buffer

که توسط Dos مدیریت می شود.

فرق بین مسئله redinet > ورودی

< خروجی ; c = get char ( )

char c ; put char (character variable)

: Test < A. dat

put char (c) ;

اگر این دو دستور در یک حلقه بکار روند می توان محتوای خطی از کاراکتر را خوانده یا چاپ نمود.

تابع scanf

scanf (control string , argl , avg 2, ……. , argn)

control string شامل اطلاعات فرمت ورودی برای arg ها می باشد و از مجموعه ای از کاراکتر ها ی کنترلی تشکیل دشه است بگونه ای که می بایست برای هر arg یک مجموعه ازاین کاراکتر های کنترلی بیان شوند.

هر مجموعه کاراکتر کنترلی با ٪ شروع می شود . در رشته کنترلی کاراکتر های کنترلی می توانند به دنبال هم بیان شوند . یا با کاراکترهای سفید white space (blank , tabsor) از هم جدا شوند . آرگومانها می بایست با L شروع شوند ولی برای اسم آرایه ها چنین نیست

کاراکترهای کنترلی که مورد استفاده قرار می گیرند به شرح زیر می باشند .

Conversion character

C data item is a single char

d data item is a decimal integer

e data item is a floating paint value با توان e

f data item is a floating paint value با ممیزاعشار

g data item is a floating paint value با توان e یا ممیز اعشار

h data item is a short integer

i data item is a decimal , her , or actal integer

o data item is a octal integer

s data item is a string followed by a whiespace char

u data item is a unsigned decimal integer

x data item is a hexadecimal integer

{….} data item is a string which may inkwell white space

```
# include < stdio . h >
main ( )
{
char item { 2. } ;
int pair num ;
```

```
float cast ;
:
scanf ( " % s % d % f ", & item , & portnum , & cost ) ;
:
}
```

Book b 123 b 1.04

Book 1230

1.04

Book 1230

1.04

اگر یک رشته با دستور scanf خوانده شود کاراکترهای سفیدw.s را نمی توان خواند در این صورت یا می بایست از توابعی چون get char( ) استفاده کرد یا در فرصت مربوطه از {... } استفاده کرد.

```
# include < stdio . h >
main ( )
{
char  line { 80 } ;
:
scanf ( " %{AB -  -  z } " ,& line ) ;
:
}
```

یک رشته با طول حداکثر 80 که می تواند شامل کاراکتر ها باشد خوانده می شود .

به محضر اینکه در ورودی به کاراکتر برخورد کنیم که داخل { ... } وجود ندارد.

بقیه قطع می شود.

NEW YORK CITY ⇨ line { } = NEW YORK CITY

NEW YORK CITY ⇨ line { } = N

دستور با فرمت "{h / ^ } ٪ " نوشته شود به این مفهوم است کاراکترها داخل { } که به دنبال ^ آمده اند نباید خوانده شوند و ورود این نوع کاراکترها به منزله انتهای رشته خواهد بود.

مشخص کردن طول فیله ها

مجموعه ای از کاراکترهای مجاور پیوسته که شامل کاراکترهای سفید نباشند میدان field نامیده می شود .

طول کاراکترهای ورودی برای یک field می تواند ازمیزان مشخص شده باشد ولی اگر بیش از آن باشد خوانده نخواهد شد.

```
# include < stdio. h >
main ( )
{
int a, b , c ;
:
scanf ( " %3 d % 3 d % 3d " , &a , &a , &a  ) ;
:
}
```

1 2 3 a = 1 b = 2 c = 3

123 456 789 a = 123 b = 456 c = 789

123456789 a = 123 b = 456 c = 789

12 34 5678 9 a = 123 b = 4 c =567

```
# include < stdio. h >
main ( )
{
int I ;
float x ;
char c ;
:
scanf ( " %3d % 5f % c " , &; , &x , &c ) ;
:
}
```

l Q 2560875 T ; = 1Q x=25608 c = '7'

اگر بین کاراکترهای کنترلی ، کاراکتر های سفید باشد این کاراکترها خوانده شده ولی دور ریخته می شود یاحتی هر کاراکتر غیر کنترلی نوشته شود خوانده شده ولی دور ریخته می شود.

```
# include < stadio. h >
main ( )
{
char c1 , c 2 , c 3 ;
:
scanf ( " % c % c % c " , & c1 , & c2 , & c3 );
:
}
```

a b c c1 = a c 2 = b c 3 = b

فقط نیاز به یک کاراکتر دارد پس کاراکتر جداساز محسوب نمی شود یک راه حل استفاده از 16٪ به جای %c می باشد راه حل دوم ورودی بدون فاصله به ورم abc و راه حل سوم scanf (" %c %c % c " , & c1, & c2 , & c3);

```
#Include < stadio .h>
Main ( )
{
int I;
float x ;
:
scanf ( " %d %f " , & I, & x);
:
}
```

خوانده شده ولی دور ریخته می شود.

1 a 2.0 ; =1 x=2

1 2.0 i=1 x=2

تابع printf

Printf (control string, arg1, arg 2,…, argn)

Arg ها بر خلاف scanf با & شروع نمی شوند.

Control stringمثل تابع scanf شامل اطلاعات فرمت میباشد.

```
# include < stadio .h>
# include < conio .h>
main ( )
{
```

```
char item { 20 } ;
int partrum ;
 float cost ; clrscr( )
 scanf ( " %s % d %f " , item . & partnum, &castl);
 printf ( " %s66 % aba %f " ,  item , copart rum , cast);
 Book b  123 b  104
 Book b b  123 b b 104
```

تابع

نوشته شود نتیجه به فرم خواهد بود.

```
 Book 123 104
 # include < stdio .h>
 # include < conio .h>
main ( )
 {
 double x = 5000 .o,y = 0.0025 ;
 clrscr (  ) ; "
 printf ( " % f b  %fb %f / n/n" , x*y );
 printf ( " %9 b%eb %e" , x,y  ,x*y );
 }
```

نتیجه 5000. 000000 b 0.002500 b 12.500000

5000000 c +003 b 2.500000 c -003b 1.250000 c +001

حداقل طول یک field را می توان در رشته کنترلی مشخص کرد اگر طول لازم برای چاپ بیش از این مقدار باشد اتفاقی نمی افتد ولی اگر کمتر باشد به سمت چپ مقدار مورد نظر b اضافه خواهد شد.

```
 # include < stdio .h>
 # include < conio .h>
main ( )
 {
 int ; = 12345 ;
 float x = 345.678;
  clrscr ( )
 printf ( " % 3 d  %5d %8d / n/n" , ; ,; ;  );
 printf ( " %3f b %7.2fb%8.1f" / n/n" , ; ,; ; );
printf ( " %10.3fb %7.2f b %8.1f" , x,x,x ) ;
}
12345b 12345bbbb 12345;
 ---
345.678bbbb345.678b … b 345.678
---
bbb 345.678 bb 345.67bbbb 345.6
```

علاوه بر مشخص کردن حداقل طول برای یک filed می توان دقت precision را نیز مشخص نمود که این عمل برای اعداد ممیز شناور بیانگر تعداد ارقام بعد از ممیز و برای رشته ها ماکزیمم تعداد کاراکترها خواهد بود.

```
 #include < stdio .h>
 # include < conio .h>
main ( )
{
float  x = 123.456;
clrscr( )
 printf ( “ %3f b %7.2fb%8.1f” / n/n” x , x,x );
printf ( “ %10.3fb %7.2f b %8.1f” , x,x,x ) ;
}
12345b 12345bbbb 1235;
 ---
345.678bbbb345.678b … b 1.236+QQ 2bbbbb 1.2 e +QQ2
# include < stdio .h>
main (  )
{
char line { 12 } = hexadecimal
:
printf ( “ % 102 % 156%1505 6%56%,line . line , line , ;
}
hexadecimal bbbb hexadecimal b…..b hexad hexad
```

flag : می بایست بلافاصله پس از علامت ٪ ظاهر شوند.

جای خالی بعد از data اضافه می شود . علامت اعداد چه + و چه – نوشته می شود .

– data item left y ustified wilhin the field

به جای فضاهای خالی اول liading blankyصفر قرار داده می شود .

قبل از اعداد علامتدار مثبت b به جای + قرار می گیرد.

سبب می شود اعداد مبنای 8 و 16 با Q x , Q شروع شوند.

سبب می شود علامت معینی حتی برای اعداد لامل هم اضافه شود .

در فرمت g مانع از حذف صفرهای سمت راست می شود.

```
#
X , C ;
E,g , f  ; ( 12 . 120)
# include < stdio . h >
main , ; {
int I = 123 ;
float
x= 12. Q . y = -3.3;
printf ( " %- + 60 % 7 , f % # 7 . f % 7g %#7g. ; . x. x, y , y )
}
+ 123 b b    bb b b b b 12 b b b b 12. b b b -3.3 – 3.3 QQQ
```

تابع eAs برای خواندن یک رشته که می تواند شامل کاراکترهای سفید هم باشد .

دارا ورودی باید به cR ختم شود.

تابع puts برای چاپ محتوای یک رشته کاراکتری بکار می رود.

```
 #include << stdio .h>
 # include < conio .h>
main ( )
{
char line { 80} ;
```

```
clrscr ( ) ;
gets ( line ) ;
puts ( line )
```

برنامه دارای تأیید بر یکدیگر interactive

```
  # Include <stdio .h>
 # include < conio >
main ( )
{
char name {20} ;
float  score 1 , score 2 ,score 3 , avg ;
printf (  " please enter your name :" );
scanf ( "%{ 8 n }" , name );
prinrf ( "please enter the fiest score :" );
scanf (" %f" , &score 1 );
:
:
avg = ( score 1 + score 2 , score 3 );
printf ("/ n /n Name : %-s /n/n" , name);
printf ("score 1 : %- 5.1f /n" ,score);
:
printf ( "Average : % - S. 1f /n/n", avg );
```

همه عبارتهای کنترلی بر اساس تصمیم گیری بر روی یکسری عبارتهای منطقی و نتیجه ناشی از آنها مجموعه ای از دستورات (یا یک دستور) را می توانند انجام دهند

و بدین طریق می توان در برنامه ها تصمیم گیری نموده ، حلقه را ایجاد نمود و بطور کلی روند اجرای برنامه ها را تغییرداد.

While statement تا زمانی که درست است اجرا می شود.

While (expression) statement

While (x>Q) ++I;

While (x<Q: : y > Q) {

; = j +k

x=x+z

- - j

} عبارتهای مرکب به ; ختم نمی شوند.

در while ابتدا عبارت منطقی مورد بررسی قرار گرفته و در صورت صحت آن جلسه ساده یا مرکب اجرا می شود .

Do while statement

Do statement while (expression)

همانند عبارت while است ولی شرط در انتها مورد بررسی قرار می گیرد.

Do print (" %d/n , digit) ; X=x-y*z

While (x> Q); ;=j-k

}

while (i< Q) در while عبارت while به ; ختم می شود.

For (expression 1 ;expression 2 expression 3

expression 1 ;

while expression 2) }

statement

```
expression 3 ;
}
```

در اجرای for برای اولین مرتبه exp 1 فقط برای یک بار اجرا شده و تا زمانی که شرط exp2 برقرار است عبارت مورد نظر (statement) اجرا شده و پس از هر بار اجرای آن exp3 نیز یک بار اجرا شده و حلقه از نو شروع می شود .

```
for ( i=Q; ; < = 10; ++I )
{
x=y+z;
printf ( " %f ",x ) ;
}
```

;=Q -1

اگر i<=10 نیست

2- اگر ;<1Q برو به قدم 7

x=y+z -3

4- x را چاپ کن

;=;+1 -5

6- برو به قدم 2

-7

For (i= 1Q , i> s ,-- i) x * =2;

توضیح کلی در رابطه با نیاز و عدم نیاز به وجود ; در عبارتهای ساده و مرکب

برنامه ای بنویسید که اعداد از Q تا و را در زیر هم و هر عدد را در یک سطر چاپ کنید.

```
# Include <stdio .h>
# include < conio.h >
main ( )
{
int sign = Q ;
clrscr ( ) ;
while ( digit < = 9 )
{
printf (“ % d/n” , digit ) ;
++digit ;
main ( )
{
int digit = Q
clrscr ( ) ;
do
printf (“ % d/n” digit ++) ;
while ( digit <= , >) ;
}
main( )
{
int digit ;
clrscr ( ) ;  digit
for ( digit = Q; digit= Q; digit < = 9; digit )
```

```
printf ( “ d/n “ , digit ) ;
}
# Include <stdio .h>
 # include < conio.h >
# define  Eol , / n ,
main (  )
{
 char letter { 80};
int tag , count = Q;
while (  ( letter { count } = get cher ( 1) ! = Eol ++ count ;
tag = count ;
  count = Q;
  while ( count ( tag)  put char( topper ( letter { count ;
{
put char ( topper ( letter { count })
++ count ;
}
{
letter { count} = get char ( _) ;
while ( letter { count } ! = Eol
{
count = count +l ;
letter { count } = get char ( ) ;
}
do    while
```

```
# Include <stdio .h>
 # include < conio .h>
# define  Eol , / n ,
main (  )
{
 char letter { 80};
int tag , count = 1;
clrscr ( ) ;
do
++ count ;
while ( letter { count } = get char (  )
tag = count ;
count = p                                   { ++ count};
letter { count } = get char ) ;
while ( letter  { count } ! = Eole
# Include <stdio .h>
 # include < conio.h >
# define  Eol , / n ,
main (  )
{
 char letter { 80};
int tag , count =;
clrscr ( ) ;
```

```
for ( count = Q ;
tag = count ;
for ( count = ; count < tag ; ++ count )
put char ( topper ( letter { count } ))
}
```

تفاوت بین do while , while , for

For برای مواردی که شرط کاملاً مشخص است و بصورت یک عدد می باشد مناسبتر است

do while , while که تعداد تکرار مشخص نیستند مناسب می باشند.

While در حالتی که شرط قبلاً خوانده شده است . و موجود است مناسب می باشد.

do while برای حالتی که شرط در حین کار خوانده می شود.

محسابه میانگین n عدد (n و اعداد باید از ورودی خوانده شوند.)

```
# Include <stdio .h>
# include < conio.h >
# define  Eol , / n ,
main ( )
{
 int n . count ;
flout  X , average , sum = ;
printf ( “How many numbers ?” );
scanf ( “ %d “ ,&n );
for ( count = 1 , count <= n; ++count )
```

```
{
printf ( " x= " );
scanf ( " 7.f " , &x) ;
sum + = x;
}
average = sum / n ;
printf ( " /nthe average  is % f in" , average ) ;
}
```

اجرا

```
How many numbers ? 3
X= 1
X =2
X =3
```

حلقه های تودرتو محاسبه میانگین چندین لیست

```
 The average = 2.0000000
NESTED loops
# Include <stdio .h>
 # include < conio.h >
# define  Eol , / n ,
main ( )
{
 int n . n, count , loops , loop count  ;
flout X , average , sum = ;
clrscr ( ) ;
printf ( " How many  lists ? " ) ;
scanf (" % d" ,  & loops ) ;
for ( loop count = 1 ; loop count <  = loops ; ++ loop count )
```

```
sum = o
print f ( " in the average is % f / n " , average ) ;
}
}
if expression statement
if ( x< 0 ) print f (" % f " , x ) ;
if ( i<0)
{
printf ( " %f" , x );
x= x-y ;
}
if ( ex p1 )  state 1 else  state 2
if ( flag ) average = sum/ n
else
average = fahs( sum ) / n;
if el if e 2 s1
else      s 2
else if e 2    s 1
else s 2  = if  e 1
{
if    e 2      s 1
else s 2
}
if e 1 }
if e 2       s 1
```

else به آخرین عدم تطابق بر می گردد.

```
{
else    s 2
if ( flag 1 > 0 )
if ( flage 2 > 0 ) .0
s 1= 0;          s 2 = 0        flage 2 <0, f;age 1                اگر
else        s 2 = 0;
if ( flag 1 > 0 )
if ( flag 2 > 0)       s 2= 0  flag 2 <0, flag 1 > 0          اگر
if ( flag 1 > 0 )
{
if ( flage 2 > 0)
s 1 = 0;
else       s 2 = 0;
}
if ( flag 1 > 0 )
}
if ( flag 2 > 0 )
s 1 = 0 ;'
}
else       s 2=0 ;
s2 = 0                                          s2 = 0  اگر flage1 انگار
ab
A    B                  z                    کاراکترها ی دیگر به " 0" تبدیل می شوند.
 # Include <stdio .h>
 # include < conio.h >
```

```
 main ( )
 {
char  line { 80}
int count ;
printf ( " % { n/8}" , line of text below ) ;
scanf ( " % { n/8}" , line ) ;
for ( cont = 0 , line { count } ! = / 0; + + count )
{
if ( ( ( line { count } > = 0 ) && ( line  { count } <' ,' ) ) ||
( ( line { count } > = A ) && ( line  { count } <,z) ) ||
( ( line { count } > = a ) && ( line  { count } <,z) )
put char ( line { count } + 1 ;
else if ( line { count } = = ' 9 ' ) put  char ( '0' ) ;
else if ( line { count } = = ' z ' ) put  char ( 'A' ) ;
else if ( line { count } = = ' z ' ) put  char ( 'a' ) ;
else put char ( '0' ) ;
```

برنامه ای بنویسید که دو عدد صحیح m , n را به عنوان ورودی خوانده و جدول ضرب $m \times n$ رابا فرم زیر چا پ کنید.

	1	2	3	4
N = 3	1	2	3	4
	2	4	6	8
	3	6	9	12

برنامه ای بنویسید که یک تاریخ هجری شمسی را با فرمت 1375/1/15 دریافت کرده

و بگوید که این تاریخ چند شنبه است و نتیجه را چاپ کند.

برنامه ای بنویسید که یک عدد صحیح را از ورودی خوانده و با تشکیل مجموع ارقام آن تشخیص دهد که بر 9 قابل قسمت است یا نه و نتیجه را با پاسخ مناسب اعلام نماید.

برنامه ای بنویسید که ضرایب یک معادله درجه دوم $ax^2 + bx + c = 0$ E(c, b, a) دریافت کرده و ریشه های آنرا محاسبه و چاپ نماید.

حالتهای مختلف با پیامهای مناسب اعلام شوند.

Switch statement

Switch (expression) statement

در حالت کلی یک عبارت مرکب که هر کدام می توانند به فرم

```
Case exp1 :
        Case exp 2 :          switch /9 choice = get char ( ) {
:
stat 2                  case ‘ r ‘:
:               case ‘R’:
stat n
printf ( “ RED” ) ;
Break;
Case ‘ w ‘;
Printf ( “ WHITE” ) ;
Printf (“ BLUE’’) ;
Break ;
}       }
switch ( choice = topper ( get char( ) ))
```

```
{
case ‘R’ :
print f ( “ R ED” ) ;
case ‘ w ‘
printf (“WITHE’’) ;
break ;
case ‘ B’ :
printf ( “ BLUE’’ ) ;
Break ;
Printf ( “Error”) ;
}
```

کنترل اجرای برنامه به خارج از حلقه

حلقه می تواند یک حلقه switch باشد یا هر نوع حلقه دیگری

```
 # Include <stdio .h>
 # include < conio .h>
 main ( )
 {
 float x , sum = Q;
clrscr ( );
  scanf ( “ % f “  , & x ) ;
    while ( x > d )
    {
    sum = sum + x ;
    if ( x> 10000)
      {
```

```
printf ( “Error – out of rang “) ;
break ;
}
}
for( count = 1 ; count < = 100; + + count )
}
scanf ( “ % f “ , &X) ;
if  ( x<0)
}
printf ( “ Error – NEGATIVE value for x”) ;
 break ;
}
  :
  }
```

حلقه های چند گانه break سبب خروج از داخلی ترین حلقه میشود .

نزدیکترین حلقه به Break

```
  For ( count = Q ; count < = n ; ++ count )
  {
  :
  while ( (get char ( ) ) = ‘ ( n )
  {
  if  ( ( = ‘ * ‘ | break ) ;
  :
   }
```

سبب اتمام حلقه while می شود.

Continue statement

عبارت Continue مانع از اجرای دستورات باقی مانده با حلقه می شود و حلقه به ازای مقدار جدید شمار ند ادامه می یابد.

```
Do {
Scan f ( " %f" , &x);
If ( x< 0)
{
printf ( " Error – NEGATIVE  VALVE FOR  X " ) ;
continue ;
}
process the connegative value of x * 1
}
while  (get char (  ) ! = 'n' )
{
if ( ( = ' *'| break)) ;
;
{
```

سبب اتمام حلقه while می شود .

Continue statement

عبارت Continue مانع از اجرای دستورات باقی

مانده حلقه می شود و حلقه به ازای مقدار جدید

شمارند ادامه می یابد.

```
Do {
scanf (“ %f , “ , &x) ;
If ( x<0)
{
printf (“Error – NEGATIVE VALVE value of x*/”);
}
while ( x<=100);
```

محاسبه میانگین اعداد مثبت یک نیست .

```
# Include <stdio .h>
# include < conio.h >
main ( )
{
 int n . n, count , navg =0  ;
flout  X , average , sum = 0;
clrscr ( ) ;
printf ( “ How many  number s ? “ ) ;
scan f (“ % d“ ,& n) ;
for ( loop count = 1 ; count <  = n; ++ count )
{
print f ( “x=”):
scanf ( “ %f “ , & xl) ;
```

```
if ( x< 0) continue ;
sum + = x;
++navg ;
}
average = sum / navg ;
printf ( " /n the average is %f /n " , average ) ;
}
```

n=6 ,x= 1 , -1 , 2 , -2,3,-3,

n=6 , x= 1,2,3,4,5,6,

آزمایش شود .

Comma operator

For (expression a , exp/b , exp 2 , exp 3 a , exp 3b)

برنامه ای برای جستجوی کلمات یا منتهای palindromes (مقلوب مستوی) در این برنامه فرض بر این است که اگر ابتدای متن وارد شده END باشد به معنی انتهای کار باشد.

این برنامه را بدون استفاده از comma operator بنویسد.

```
# Include <stdio .h>
# include < conio.h >
# define  EOL '/n '
#define TRVE 1
# define FALSE
 main ( )
{
char letter { 80 } ;
```

```
while ( 1 )
{
flag = TRUE ;
 prinrf ( " please enter a text " ) ;
for ( cont =  ( letter count) ! = EoL) ;
++ count ;
if (letter count ! = letter { count back } )
{
flag = FALSE ;
break ;
}
for ( count = 0; count < = tag ; ++ count )
put char ( letter { count } ) ;
if ( flag )
print f (" ISA palindromes ( n / n) " ) ;
else
print f ( " Is NOT% a palindromes ( n / n ) " ) ;
}
```

اکثراً برای کنترل اجرای برنامه از داخل حلقه های تودرتو به بیرون بکار میرود.

```
Go to lahali label : statement
Scanf ( " % f " ,& x) ;
While ( x< = 100 )
{
:
if ( x < 0 ) go to error check ;
```

:

}

error char : {

printf (" ERROR – NEGATIVE VALUE FOR x ") ;

برنامه palind rones را بدون استفاده از comma بنویسید. }

برنامه کاملی بنویسید که با دریافت ضرایب x_i, f_i, n میانگین وزنی و هندسی را از فرمولهای زیر برای n عدد محاسبه نماید .فرمت مناسبی برای ورودی انتخاب کنید.

میانگین وزنی wegh ted average $xavg = f_1 x_1 + f_2, x_2 , +.... f_n,x_n$

میانگین هندسی geometric avetage $Xavg =[x_1 ..... x_2] 1/n$

نسری فیبو ناجی توسط رابطه زیر مشخص می شود .

$F_i = f_{1-1} + f_{i-2}$

$F_{1} = f_2 = 1$

برنامه ای بنویسید که با دریافت عدد n از این سری را محاسبه و چاپ نماید.

با استفاده از حلقه های تودرتور و بدست آوردن یک فرمول برنامه ای بنویسید که هرم اعدادی به فرم زیر را چاپ کند/.

```
          1
        2 3 2
      3  4  5 4 3
    4 5 6 7 6 5 4
  5 6 7 8 9 8 7 6 5
6 7 8 9 0 1 0 9 8 7 6
```

فصل هشتم توابع

تعریف تابع defining a function

آرگومان یا پارامترهای formal یا مجازی deta – type name (arg)....

هر تابع مجموعه ای از آرگومانها را می تواند داشته باشد ولی فقط یک مقدار برگشتی دارد تعریف تابع در ابتدای برنامه یا انتها می تواند صورت گیرد که انتهای برنامه معمول است.

```
Lower – to – upper ( c )
Char c 1 ;                    تعریف آرگومانهای تابع
                              تعریف متغیرهای محلی    char c2;

C2 = ( c1> = 'a' && c1 <='z') ? (A+c1-'a') : c1;
Return ( c2 ) ;
```

بحث کلی در رابطه با آرگومانهای مجازی مشخص کننده مقدار برگشتی تابع

آرگومانهای تابع دو جا می تواند تعریف شود.

```
Lower – to – upper ( char c 1)
{
char c2 ;
if (c1 >='a' &&c1 < = 'z' )
return ( 'A' + c1 – 'a');
else
return ( c1);
}
```

اعلان تابع function delaratio نوع تابع نیز می بایست مثل نوع متغیرها در ابتدای برنامه و همراه اعلان متغیرها اعلان گردد که اگر این کار صورت نگیرد تابع از نوع in+ فرض می شود .

تابع محاسبه فاکتور یل n!

```
Long int factor ( int n )
{
```

```
int     i;
long int prod = 1;
if ( n<1) for ( i=2; i<=n ; ++i) prod * = I;
```

نوع مقدار برگشتی با نوع تابع یکسان است.

پیدا کردن بزرگترین عدد بین دو عدد صحیح

اگر چه لزومی ندارد ولی بهتراست.

```
Int max ( int +x , int y)
{
int z;
z = ( x>y ) ? x:y;
return ( z) ;
}
```

اگر پس از return چیزی نباشد تابع مقداری را به برنامه صدازننده برگشت نمیدهد.

دسترسی به تابع (صدا زدن) accessing a function

برای صدا زدن یک تابع کافی است نام تابع را به همراه آرگومانهای آن که در داخل پارانتز قرار گرفته مشخص نماییم . ارگومانهایی که به هنگام صدا زدن یک تابع بکار می روند آرگومانها یا پارامترهای حقیقی می باشند این نحوه دسترسی یا صدا زدن را دسترسی یا صدا زدن با مقدار می نامیم چون آنچه که منتقل می شود مقدار آرگومانها ست نه خود آنها . یک تابع را از هر نقطه ای می توان صدا زد.

مثال : برنامه ای برای پیدا کردن max سه عدد

```
# Include <stdio .h>
```

```
 # include < conio.h >
 main (  )
 {
  int a . b, c, d   ;
int max ( int , int ) ;
print f ( “ /na=” );
scan f ( “ %d “, &a) ;
printf ( “ /nb = “ ) ;
scanf ( “ %d” ,  &h) ;
printf ( “ inc = “ ) ;
scan f ( “ %d “ , & c ) ;
d = max ( a , b ) ;
printf ( “ /n/n maximum = % d” , max ( c , d ) ;
{
int max ( int x , int y )
{
int z;
z = ( x > = y ) ? x : y ;
return ( z) ;
}
```

همانگونه که بیان شد انتقال پارامترها به تابع به صورت مقدار است یعنی با صدا زدن مقدار آرگومانهای حقیقی به داخل تابع کپی می شود . مقدار آرگومانهای مجازی در داخل تابع می تواند تغییر کند ولی مقدار آرگومانهای واقعی متناخر در برنامه صدا زننده چیزی خواهد کرد.

```
# Include <stdio .h>
main ( )
{
 int a . 2;
int modify  ( int )
printf ( " /n a = %d ( from main after calling the function ) " , a) ;
}
int modify ( int a)
}
```

اگر چه اسم مشترک دارند دو متغیر کاملاً جدا هستند.

```
Print f ( " /n/n a = %d ( from the function after being modified ) ", a);
```

چون مقدار a برگشت داده نشده مقدار آرگومان واقعی دربرنامه

```
Return ;
```

صدا زننده تغییر نمی کنند.

```
a=2
a=6    recursive  function      , recursion
a=2
```

توابع برگشت پذیر

```
N ! = n* ( n – 1) ;
}
```

نحوه اجرای توابع recursive توضیح داده شود.

تا زمانی که شرط انتها فرا نرسیده چیزی اجرا نشده و همه چیز ثبت می شود و پس ا

رسیدن شرط خاتمه توابع به صورت عکس اجرا می شوند.

تابعی برای دریافت و چاپ یک text بصورت عکس

```
# Include <stdio .h>
```

```
# include < conio.h >
main ( )
{
void erverse ( void ) ;
  printf ( please enter a line of text below / n " ) ;
 reverse ( ) ;
 }
 void recerse ( void )
 } char text
 if ( ( text – get char ( )) ;= EOL) reverse ( ) ;
 put char ( text ) ;
 return ;
 }
 void reverse ( void )
 {
 char text {80};
 int count , tag ;
 for ( cont = 0;( (text { count } = get char ( ) != EOLN) ;++
 tag = count – 1;
 for ( count = tag ; count> = 0; - - count )
 put char ( text { count} ;
 return  ;
 }
```

Learn the towers of Hano I برج های نوری

```
# Include <stdio .h>
main (  )
{
void transfer ( int , char , char , char , ) ;
int  n ;
printf ( “ welcome to the towers of HAMI ( n )  temporary”);
printf ( “ %HOWmany disks ? “ ) ;
scanf ( “ %d “, & n ) ;
transfer ( n, L ‘ , R , c , ) ;
}
void transfer ( int n, char from char to , char tomp )
{ if ( n>01)
}
Transfer (1) -1, from, temp, tol);
printf ( move disk %d from % c to % c /n “ , n , from , to )
transfer  ( n-1 ) temp , to , from ) ;
}
return ;
{
```

صدا زدن تابع با $n = 3$, L; R, C

مفهوم return و برگشت به نقطه صدا زدن یا انتقال مقدار یا بدون آن توضیح داده شود.

برگشت پذیر برنامه ای بنویسید که با دریافت اعداد x,n مقدار عبارت s_n z1-x+ x^2 /2 –x^3 /6….. را محاسبه و چاپ نماید n عدد صحیح و x عدد اعشاری است با استفاده ا ز توابع برگشت پذیر برنامه ای بنویسید که با دریافت اعداد n و x (n صحیح (x عدد اعشاری n(-1<x<1 جمله اول چند جمله ای لژ اندر را محاسبه و چاپ نماید.

P0 = 1

P 1 = x

P n = { (zn – 1) / n } * p n – 1 – { (n-1)/n } – pn – 2}