برنامه‌سازی کامپیوتر

جلسه سوم

آشنایی با زبان C

طرح کلی

- تاریخچه زبان C
- آشنایی با ویژگی‌های اصلی زبان C
- انواع داده‌ای در زبان C
- ثوابت و متغییرها
- عملگرها
- تقدم عملگرها
- تبدیل انواع

تاریخچه

- در سال ۱۹۷۲ توسط دنیس ریچی و با تکامل یافتن زبان BCPL (که طراح آن مارتین ریچاردز است) برای برنامه‌نویسی سیستم‌ها طراحی شد.
- زبان BCPL از زبان B که طراح آن کِن‌تامپسون است گرفته شده است.
- علت نام‌گذاری زبان C به این دلیل است که بعد از B طراحی شده است.
- زبان C++ نیز از C مشتق شده است و مفاهیمی همچون شئ‌گرایی و کلاس‌ها را به زبان C اضافه می‌کند.
- تا سال ۱۹۷۸ فقط در آزمایشگاه بل مورد استفاده قرار می‌گرفت و در این سال بصورت عمومی منتشر شد.
- نسخه‌ی استانداردی از آن توسط ANSI برای جلوگیری از ناسازگاری و حفظ قابلیت حمل تهیه و منتشر گردید.

ویژگی‌های زبان C (۱)

- زبان C یک زبان میانی است. چون هم مانند زبان‌های سطح پائین مانند اسمبلی، مستقیماً به حافظه دستیابی داشته و با مفاهیم بیت و بایت و آدرس کار می‌کند و هم مانند زبان‌های سطح بالایی چون پاسکال، دستورالعمل‌های آن به زبان طبیعی نزدیک بوده و قابلیت خوانایی بالایی دارد.
- زبان C زبانی ساخت‌یافته است. یعنی دارای ساختارهایی همچون شرط و تکرار است که قابلیت خوانایی برنامه‌ها را بالا برده و درک آن‌ها را آسان‌تر می‌سازد.
- زبان C بسیار منعطف و قدرتمند است.
- زبان برنامه‌نویسی سیستم است.
- رابطه تنگاتنگی با اسمبلی دارد و تمام قابلیت‌های اسمبلی را می‌توان در آن استفاده کرد.
- زبان قابل حمل است. یعنی روی کامپیوترهای مختلف می‌توان آن را نوشت و بدون تغییر یا با تغییر اندک آن را روی سایر کامپیوترها اجرا کرد.

زبان‌های ساخت‌یافته	زبان‌های غیرساخت‌یافته
پاسکال	فرترن
اِدا	بیسیک
جاوا	کوبول
ماجولا-۲	
C، (C++)	

زبان‌های سطح بالا	زبان‌های میانی	زبان‌های سطح پائین
پاسکال	جاوا	ماکرواسمبلر
اِدا	فورث	اسمبلر
ماجولا-۲	C، (C++)	
کوبول		
بیسیک		

ویژگی‌های زبان C (۲)

- زبان کوچکی است. تقریباً ۳۲ کلمه کلیدی دارد. زیاد بودن کلمات کلیدی یک زبان (مثل بیسیک که ۱۵۰ کلمه دارد) نشانگر قدرت آن نیست. برخی کامپایلرها علاوه بر ۳۲ کلمه کلیدی برخی کلمات دیگر را نیز افزوده‌اند.
- نسبت به کوچکی و بزرگی حروف حساس است.
- کلمات کلیدی آن با حروف کوچک نوشته می‌شوند. while یک کلمه کلیدی است ولی WHILE نمی‌باشد.

کلمات کلیدی زبان C

auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**short**	**unsigned**
continue	**for**	**signed**	**void**
default	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**

کلمات کلیدی برخی کامپایلرها

asm	**_cs**	**_ds**	**_es**
_ss	**cdecl**	**far**	**huge**
interrupt	**near**	**pascal**	

دستورالعمل‌ها

- هر دستور به ; ختم می‌شود.
- حداکثر طول یک دستور ۲۵۵ کاراکتر است.
- هر دستور می‌تواند در یک یا چند سطر ادامه داشته باشد.

```
بخش اول دستور
بخش دوم دستور
...
بخش آخر دستور ;
```

- در هر سطر می‌توان چند دستور را نوشت. این کار توصیه نمی‌شود.

```
دستور ۱ ;دستور۲ ;دستور۳ ; ...
```

- توضیحات عباراتی هستند که اجرا نمی‌شوند و مابین دو علامت */ و /* قرار می‌گیرند و یا بعد از // ظاهر می‌شوند.

```
/* توضیحات چند خطی */
// توضیحات یک خطی
```

انواع داده‌ای

- برنامه‌های کامپیوتری دارای دو بخش اصلی **داده** و **دستور** هستند.
- زبان C دارای ۵ نوع داده اصلی است: char، int، float، double و void.
- مقادیر انواع داده‌ای مختلف در پردازنده‌های مختلف ممکن است متفاوت باشد.
- علاوه بر این پنج نوع، می‌توان با افزودن پیش‌وند، محدوده‌ی انواع را کنترل کرد.
- void نوع ویژه‌ای است که بیش‌تر در توابع مورد استفاده قرار می‌گیرد.

نوع	کاربرد	پیشوندها
char	**انواع کاراکترها و سمبل‌ها مانند '%' 'A' 'c' '5'**	**unsigned و signed**
int	**اعداد صحیح**	**unsigned/signed و long/short**
float	**اعداد اعشاری**	**-**
double	**اعداد اعشاری بزرگتر از float**	**long**

انواع داده‌ها و مقادیر قابل قبول آن‌ها

نوع	اندازه به بیت	بازه قابل قبول
char	۸	۱۲۷- تا ۱۲۷
unsigned char	۸	۰ تا ۲۵۵
signed char	۸	۱۲۷- تا ۱۲۷
int	۱۶ یا ۳۲	۳۲۷۶۷- تا ۳۲۷۶۷
unsigned int	۱۶ یا ۳۲	۰ تا ۶۵۵۳۵
signed int	۱۶ یا ۳۲	۳۲۷۶۷- تا ۳۲۷۶۷
short int	۱۶	۳۲۷۶۷- تا ۳۲۷۶۷
unsigned short int	۱۶	۰ تا ۶۵۵۳۵
signed short int	۱۶	۳۲۷۶۷- تا ۳۲۷۶۷
long int	۳۲	۲۱۴۷۴۸۳۶۴۷- تا ۲۱۴۷۴۸۳۶۴۷
signed long int	۳۲	۲۱۴۷۴۸۳۶۴۷- تا ۲۱۴۷۴۸۳۶۴۷
unsigned int	۳۲	۰ تا ۴۲۹۴۹۶۷۲۹۵
float	۳۲	۷ رقم دقت (ارقام بعد از اعشار)(تقریباً $۱۰^{-۳۸}$ تا $۱۰^{۳۸}$)
double	۶۴	۱۵ رقم دقت (تقریباً $۱۰^{-۳۰۸}$ تا $۱۰^{۳۰۸}$)
long double	۸۰	۱۹ رقم دقت (تقریباً $۱۰^{-۴۹۳۲}$ تا $۱۰^{۴۹۳۲}$)

متغییرها

متغییر ظرفی برای ذخیره‌ی مقادیر است که در طول اجرای برنامه مقادیر مختلفی ممکن است در آن قرار گیرد. در کامپیوتر متغییر نامی برای بخشی از حافظه است که مقادیری از نوع خاص در آن قرار می‌گیرد.

- برای دسترسی به متغییرها از نام آن‌ها استفاده می‌کنیم. پس متغییرها امکان نام‌گذاری کلمات حافظه را فراهم می‌کنند.
- شرایط نام‌گذاری متغییرها:
  - ترکیبی از حروف ‘a’ تا ‘z’ و ‘A’ تا ‘Z’ و ارقام 0 تا 9 و همچنین علامت ‘_’ می‌باشند.
  - اولین کاراکتر نباید رقم باشد.
  - طول متغییرها محدود نمی‌باشد ولی تنها ۳۱ کارکتر اول آن مورد استفاده قرار خواهد گرفت.
- مثال: اسامی مجاز مانند count، test23، sum، S_1 و اسامی غیرمجاز مانند 1test، high!there، grade.1، pcx..

تعریف یا اعلان متغییرها

- متغییرها محل ذخیره داده‌ها هستند پس چون داده‌ها نوع دارند پس متغییرها نیز دارای نوع هستند. تعیین نوع متغییر را تعریف متغییر یا اعلان آن می‌گوئیم.
- قبل از استفاده از یک متغییر باید نوع آن را مشخص کرد.
- نوع متغییر، مقادیری را که متغییر می‌تواند بپذیرد و اعمالی را که می‌توان روی آن مقادیر انجام داد را مشخص می‌کند.

نوع داده نام متغییر ;

نوع داده نام متغییر۱, نام متغییر۲, ...;

```
int        x,y;
float      m,n;
char       ch1,ch2;
double     d1;
long int   p1;
```

مقدار دادن به متغییرها

- **هنگام تعریف متغییر (تعیین نوع آن)**

```
int    x , y = 5;
char    ch1 = 'a' , ch2 = 'm';
```

- **پس از تعریف متغییر و با دستور انتساب (=)**

```
int          x, y, m ;
float        f1, f2 ;
char         ch1, ch2 ;
f1 = 15.5 ;
f2 = 20.25 ;
x = y = m = 0 ;
ch1 = ch2 = 'a' ;
```

نکته: مقداردهی چندگانه در C امکانپذیر است.

- **دستورات ورودی**

```
int  x , y ;
scanf ("%d %d", &x , &y ) ;
```

ثابت‌ها

- مقادیری هستند که در طول اجرای برنامه مورد استفاده قرار می‌گیرند ولی تغییر نمی‌کنند.
- نام‌گذاری ثوابت از نام‌گذاری متغییرها پیروی می‌کنند.
- مقداری که برای ثابت تعیین می‌شود نوع آن را مشخص می‌کند.
- مثال :

عدد پی، یا مقادیری که توسط برنامه‌نویس تعیین‌شده و تا انتهای برنامه ثابت‌اند.

Pi = 3.14

Pi = 3.1415

N = 10

N =100

روش‌های تعریف ثوابت

- دستور #define

#define <نام ثابت> <مقدار ثابت>

مثال:

```
#define  pi        3.14
#define  M         100
```

در انتهای دستور ; قرار نمی‌گیرد چون دستور پیش‌پردازنده است.
پش‌پردازنده برنامه‌ای از سیستم است که قبل از ترجمه برنامه توسط کامپایلر تغییراتی را در آن اعمال می‌کند. در اینجا پیش‌پردازنده، نام ثابت را در تمام برنامه با مقدار ثابت جایگزین می‌کند.

- دستور const

const <نوع داده> <نام ثابت>=<مقدار> ;

مثال:

```
const     int  n = 100 , int count = 50;
const     signed   char   x = 'a' ;
```

عملگرها

- نمادهایی هستند که برای انجام اعمال خاصی مورد استفاده قرار می‌گیرند. بعد از تعریف متغییرها و مقدار دادن به آن‌ها باید بتوان عملیاتی را روی آن‌ها انجام داد.
  - عملگرهای محاسباتی
  - عملگرهای رابطه‌ای
  - عملگرهای منطقی
  - عملگرهای بیتی

عبارات ترکیبی از متغییرها، ثوابت، مقادیر و عملگرها هستند.

عملگرهای محاسباتی

عملگر	نام	مثال
-	تفریق و منهای یکانی	x-y یا -x
+	جمع	x+y
*	ضرب	x*y
/	تقسیم	x/y
%	باقیمانده تقسیم	x%y
--	کاهش (decrement)	x-- یا --x
++	افزایش (increment)	x++ یا ++x

افزایش و کاهش (۱)

- اگر عملگرهای ++ و – قبل از عملوند قرار گیرند، ابتدا این عملگرها عمل می‌کنند و نتیجه آن‌ها در محاسبات شرکت می‌کنند.
- اگر عملگرهای ++ و – بعد از عملوند قرار گیرند، ابتدا مقدار فعلی عملوند در محاسبات مورد استفاده قرار گرفته و سپس این عملگرها عمل می‌کنند.
- مثال:

```
int   x, y;
x = 10 ;
y = ++ x;

int   x, y;
x = 10 ;
y = x ++;
```

افزایش و کاهش (۲)

مثال:

```
int   x, y, m;
x = 10 ;
y = 15 ;
```

مقادیر متغییرها قبل از انجام عمل جمع

```
m = ++ x + y ++;   →m=26      x=11   y=15
m=x+++y++;         →m=25      x=10   y=15
m=x+++++y;         →error
m=(x++)+ (++y);    →m=26      x=10   y=16
m=++x + ++y;       →m=27      x=11   y=16
```

تقدم عملگرهای محاسباتی

بالاترین تقدم	++ --
	منهای یکانی-
	* / %
پائین‌ترین تقدم	- +

- مثال:

```
int   m, x = 6 , y = 10 ;
m = x + y / 2 * 3;          → m= 21
m = (x + ((y / 2) * 3));
```

عملگرهای رابطه‌ای

- ارتباط مابین عملوندها را مشخص می‌کنند.
- حاصل عملیات، یکی از ارزش‌های درست یا نادرست است.
- در تصمیم‌گیری‌ها و تغییر مسیر اجرایی برنامه کاربرد دارند.

عملگر	نام	مثال
>	بزرگتر	x > y
>=	بزرگتر یا مساوی	x >= y
<	کوچکتر	x < y
<=	کوچکتر یا مساوی	x <= y
==	متساوی	x == y
!=	نامساوی	x != y

عملگرهای منطقی

- بر روی عبارات منطقی عمل می‌کنند. عبارات منطقی محاسباتی را بر روی دو ارزش درستی (True یا T) و نادرستی (False یا F) انجام می‌دهند.
- در زبان C ارزش نادرستی با صفر، و ارزش درستی با غیرصفر مشخص می‌شود.

```
int x = 0, y = 4, m, p, q ;
m = x && y ;
p = x || y ;
q = ! x ;
```

عملگر	نام	مثال
!	نقیض (not)	!x
&&	و (and)	x > y && m < p
||	یا (or)	x > y || m < p

x	y	x && y	x || y	!x
F	F	F	F	T
F	T	F	T	T
T	F	F	T	F
T	T	T	T	F

عملگرهای ترکیبی

- از ترکیب عملگرهای محاسباتی و نماد = ایجاد می‌شوند. نسبت به سایر عملگرها از اولویت پائین‌تری برخوردارند.

عملگر	نام	مثال	معادل
+=	انتساب جمع	x += y	x = x + y
-=	انتساب تفریق	x -= y	x = x - y
*=	انتساب ضرب	x *= y	x = x * y
/=	انتساب تقسیم	x /= y	x = x / y
%=	انتساب باقی‌مانده تقسیم	x %= y	x = x % y

عملگرهای بیتی (۱)

- برای انجام بسیاری از کارهایی که در زبان اسمبلی قادر به انجام آن‌ها هستیم مورد استفاده قرار می‌گیرند.
- از جمله کاربردهای عملگرهای بیتی می‌توان به خواندن بیت‌های وضعیت دستگاه‌های چاپگر و مودم و تست وضعیت آن‌ها اشاره کرد.
- برای تست کردن، مقدار دادن یا شیفت دادن و سایر اعمال بر روی مقادیری که در یک بایت (char) یا کلمه (int) ذخیره شده‌اند به کار می‌روند.
- عملگرهای بیتی را نمی‌توان با انواع float، double، long double و void و یا سایر انواع پیچیده به کار برد.

عملگر	نام	
&	و	(AND)
|	یا	(OR)
^	یای انحصاری	(XOR)
~	نقیض	(NOT)
>>	شیفت به راست	(Right Shift)
<<	شیفت به چپ	(Left Shift)

x	y	x & y	x | y	x ^ y	~x
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

عملگرهای بیتی (۲)

- عملگرهای شیفت، بر روی یک عملوند عمل می‌کنند و بیت‌های آن را به سمت راست یا چپ شیفت می‌دهند. این عملگرها بصورت زیر بکاربرده می‌شوند.

متغییر << تعداد شیف ;

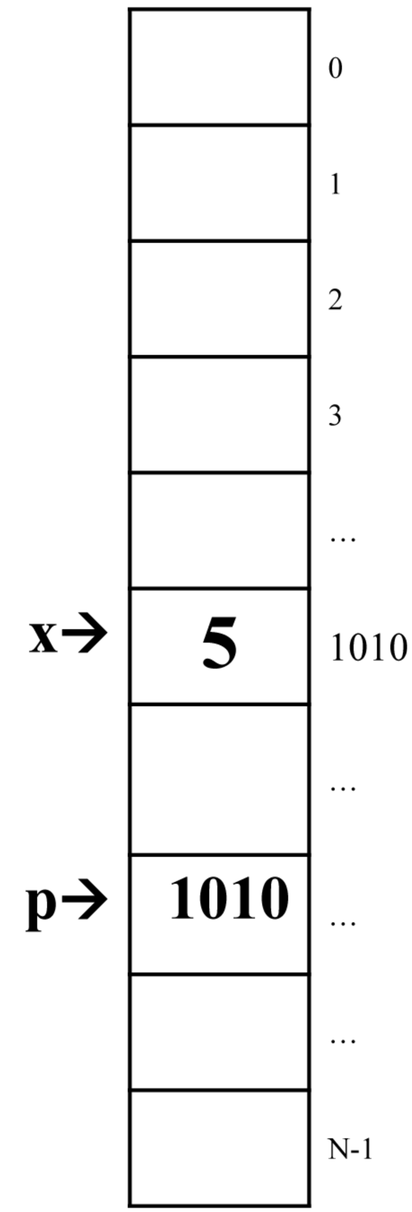
متغییر >> تعداد شیف ;

- هنگام شیفت به راست، از طرف چپ به تعداد لازم صفر اضافه می‌کنیم.
- هنگام شیفت به چپ، از طرف راست به تعداد لازم صفر اضافه می‌کنیم.
- تا زمانی که اطلاعاتی را از دست نداده‌ایم، با شیفت به راست تقسیم بر۲ و با هر شیفت به چپ ضربدر ۲ می‌کنیم.

unsigned char x ;	مقدار دودویی x	مقدار دهدهی
x = 7 ;	0000 0111	7
x = x << 1 ;	0000 1110	14
x = x << 3 ;	0111 0000	112
x = x << 2 ;	1100 0000	192
x = x >> 1 ;	0110 0000	96
x = x >> 2 ;	0001 1000	24

عملگرهای & و *

متغییرها نامی برای کلمات حافظه‌اند و کلمات حافظه نیز دارای شماره ردیف هستند که به آن‌ها آدرس می‌گوئیم. با استفاده از عملگر & به آدرس متغییرها دسترسی پیدا می‌کنیم. عملگر * نیز برای دسترسی غیرمستقیم به حافظه مورد استفاده قرار می‌گیرد.



p = & x ; آدرس x در p قرار می‌گیرد.

* p = 5 ; جایی که آدرس آن در p است (همان x)، برابر با ۵ می‌شود.

m = * p ; محتویات جایی که آدرسش در p است (۵)، در m قرار می‌گیرد.

عملگر ؟

متغییر= <عبارت۱> ? <عبارت۲> : <عبارت۳> ;

ابتدا عبارت۱ مورد بررسی قرار می‌گیرد:

✓ اگر ارزش آن درست باشد عبارت۲ برای انتساب مورد استفاده قرار می‌گیرد.

× اگر ارزش آن نادرست باشد عبارت۳ برای انتساب مورد استفاده قرار می‌گیرد.

- مثال:

```
int   x, y ;
x = 5 ;
y = x > 5  ?  x * 2  :  x * 5 ;
```

عملگر کاما (,)

- برای انجام چند عمل در یک دستور به کار می‌رود و روش استفاده از آن بدین ترتیب است:

متغییر = (<عبارت۱> , <عبارت۲>) ;

- معمولاً عبارت۱ به نحوی با عبارت۲ در ارتباط است. بعد از این‌که ارزیابی و اجرای عبارت۱ به پایان رسید، عبارت۲ می‌تواند از نتیجه آن استفاده کند.

- مثال:

```
int   x, y ;
y = (x = 2 , x * 4 / 2) ;
```

عملگر sizeof

- عملگر زمان ترجمه است.
- این عملگر می‌تواند طول یک متغییر یا نوع داده را بر حسب بایت تعیین کند.
- به دو روش زیر بکاربرده می‌شود و هنگامی که نوع داده‌ای مورد استفاده قرار می‌گیرد باید داخل پرانتز قرار گیرد.

sizeof متغییر ;
sizeof (نوع داده) ;

مثال:

```
int x, y, m ;
x = sizeof y;
m = sizeof (float) ;
```

عملگر پرانتز ()

پرانتزها عملگرهایی هستند که تقدم عملگرهای داخل خود را بالا می‌برند.

- مثال

$$y = 4 * 2 / (3 + 1) + (6 + (7 - 2))$$

$$y = 4 * 2 / \quad 3 + 1 \quad + \quad 6 + \quad 7 - 2$$

تقدم عملگرها

بالاترین تقدم	()
	! ~ ++ -- sizeof
	* / %
	+ -
	<< >>
	< <= > >=
	== !=
	&
	^
	|
	&&
	||
	?
پائین‌ترین تقدم	= += -= *= /= %=

نکته

- در عملگرهای محاسباتی که دو عملوند می‌گیرند هر دو عملوند باید از یک نوع داده‌ای باشند. مقدار حاصل از محاسبه نیز از همان نوع خواهد بود.
- مثال

3 + 5 → 8
3.0 + 5 → 8.0
10 / 4 → 2
10 / 4.0 → 2.5
10.0 / 4 → 2.5
10.0/4.0 → 2.5
(float) 10/4 → 2.5 (به این روش تبدیل نوع type casting می‌گویند)

سوال:

```
int   x = 22 , y = 14;
float    f = 13.5;
x = f + y;
```

تبدیل انواع (۱)

- **تبدیل انواع در عبارات محاسباتی**
- **قاعده کلی: انواع کوچکتر به انواع بزرگتر تبدیل می‌شوند.**
  - اگر یکی از عملوندها long double باشد، عملوند دیگر به long double تبدیل می‌شود.
  - وگرنه، اگر یکی از عملوندها double باشد، عملوند دیگر به double تبدیل می‌شود.
  - وگرنه، اگر یکی از عملوندها float باشد، عملوند دیگر به float تبدیل می‌شود.
  - وگرنه، اگر یکی از عملوندها unsigned long باشد، عملوند دیگر به unsigned long تبدیل می‌شود.
  - وگرنه، اگر یکی از عملوندها long باشد، عملوند دیگر به long تبدیل می‌شود.
  - وگرنه، اگر یکی از عملوندها unsigned int باشد، عملوند دیگر به unsigned int تبدیل می‌شود.

نکته: اگر یکی از عملوندها از نوع long و دیگری از نوع unsigned int باشد، ولی مقدار unsigned int نتواند توسط long نمایش داده شود، هر دو عملوند به unsigned long تبدیل می‌شوند.

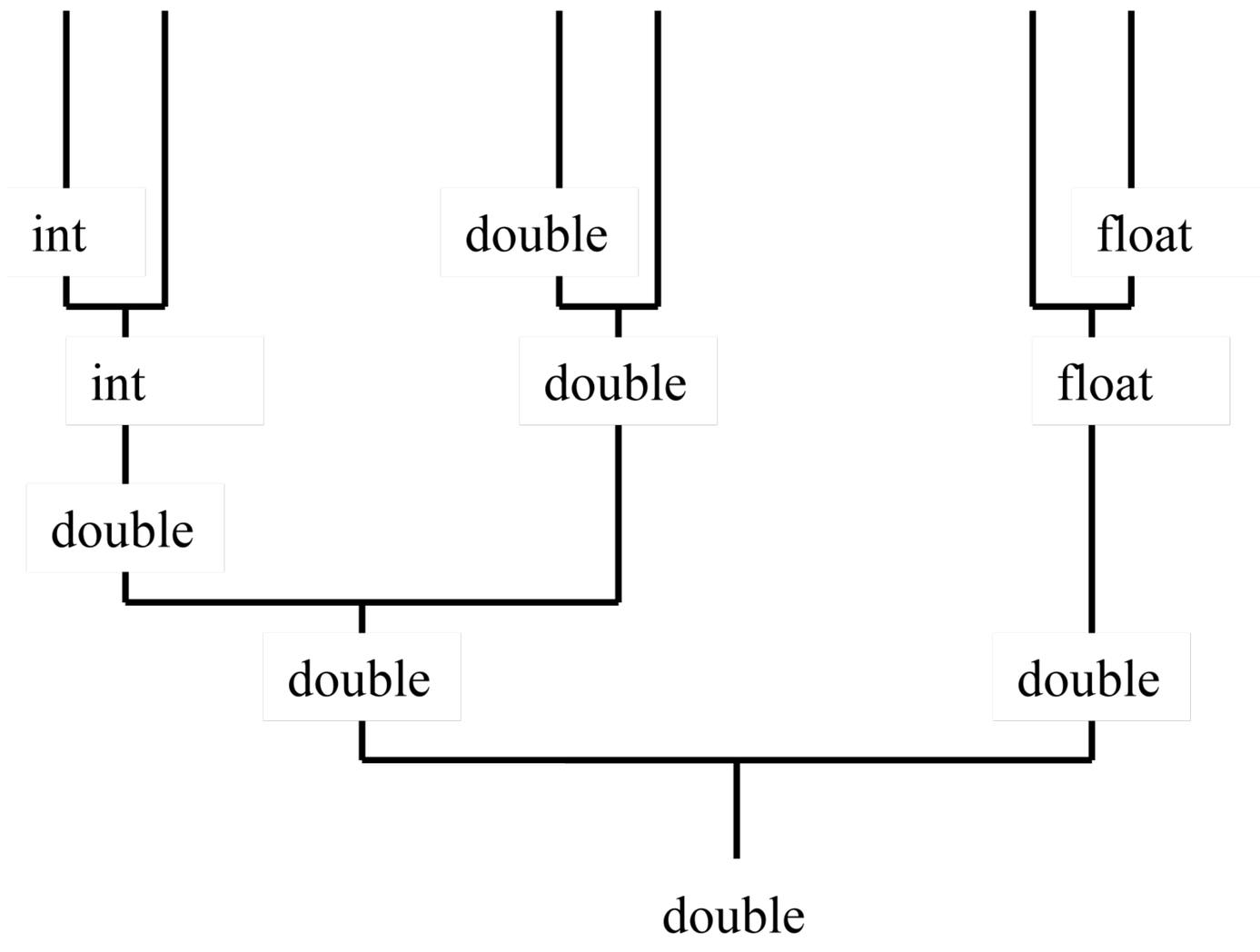
تبدیل انواع (۲)

د

- مثال:

```
char      ch ;
int       i;
float     f;
double    d;

result = ( ch / i )   +   ( f * d )   -   ( f + i )
```



تبدیل انواع (۳)

- تبدیل انواع در احکام انتساب

```
int   x ;
char   ch;
float   f;
….
ch = x ;
x = f ;
f = ch ;
f = x ;
```

- در تبدیل انواع داده‌ای به یکدیگر ممکن است بخشی از داده‌ها از بین برود.
- تبدیل نوع مستقیم double به int وجود ندارد، ابتدا double به float و سپس float به int تبدیل خواهد شد.
- برنامه‌نویس باید خود در تبدیل انواع دقت کافی داشته باشد تا اطلاعات سودمند را از دست ندهد.
- مثال: تبدیل یک int به یک char ممکن است موجب از بین رفتن بخش پرارزش مقدار int شود.

تبدیل انواع در احکام انتساب

نوع منبع	نوع مقصد	اطلاعاتی که ممکن است از بین برود
char	signed char	اگر مقدار بیش از ۱۲۷ باشد، مقصد منفی خواهد شد.
short int	char	۸ بیت با ارزش
int (۱۶ بیتی)	char	۸ بیت با ارزش
int (۳۲ بیتی)	char	۲۴ بیت با ارزش
long int	char	۲۴ بیت با ارزش
int (۱۶ بیتی)	short int	اطلاعات از بین نمی‌رود
int (۳۲ بیتی)	short int	۱۶ بیت با ارزش
long int	int (۱۶ بیتی)	۱۶ بیت با ارزش
long int	int (۳۲ بیتی)	اطلاعات از بین نمی‌رود
float	int	بخش کسری و یا بیش‌تر
double	float	دقت علائم کم می‌شود و نتیجه گرد می‌شود
long double	double	دقت علائم کم می‌شود و نتیجه گرد می‌شود

جمع‌بندی

- تاریخچه زبان C
- آشنایی با ویژگی‌های اصلی زبان C
- انواع داده‌ای در زبان C
- ثوابت و متغییرها
- عملگرها
- تقدم عملگرها
- تبدیل انواع