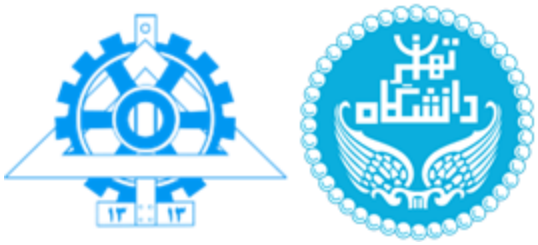

C Programming Language Documentation

طراحی کامپایلر و زبان‌های برنامه‌نویسی

بهار ۱۴۰۴

فهرست مطالب

۱) مقدمه

زبان برنامه‌نویسی C در اوایل دهه 1970 توسط Dennis Ritchie در آزمایشگاه Bell توسعه داده شد. این زبان برای توسعه‌ی سیستم‌عامل Unix طراحی شد و به سرعت محبوبیت یافت. قبل از C، زبان‌های برنامه‌نویسی مانند B و Assembly برای توسعه‌ی سیستم‌های نرم‌افزاری مورد استفاده قرار می‌گرفتند، اما C توانست امکانات بهتری را برای برنامه‌نویسان فراهم کند. این زبان در ابتدا برای بهینه‌سازی قابلیت‌های سخت‌افزاری و افزایش کارایی[1] Unix توسعه داده شد، اما به دلیل قابلیت‌های بالای آن، خیلی زود فراتر از این سیستم‌عامل گسترش یافت. این زبان به برنامه‌نویسان اجازه می‌دهد تا مستقیماً با حافظه و سخت‌افزار سیستم تعامل داشته باشند، در حالی که همچنان از امکانات برنامه‌نویسی ساخت‌یافته[2] بهره می‌برد. این زبان به عنوان جایگزینی برای زبان‌های سطح پایین مانند Assembly مطرح شد، در حالی که انعطاف‌پذیری و عملکرد[3] بالایی را حفظ کرد. در سال 1989 نسخه‌ای استاندارد از این زبان به نام ANSI C معرفی شد که یک نسخه‌ی استاندارد و پایدار از زبان C را ارائه داد. پس از آن، استانداردهای دیگری مانند C99 و C11 نیز معرفی شدند که قابلیت‌های جدیدی را به این زبان اضافه کردند [1,2].

زبان برنامه‌نویسی C یک زبان Imperative و compiled است و به صورت static type می‌باشد. کدها در این زبان پس از کامپایل موفق، به کد اسمبلی و سپس به کد ماشین تبدیل می‌شوند. این زبان به گونه‌ای طراحی شده که قابلیت دسترسی low-level به حافظه را به کاربر می‌دهد. وجود قابلیت‌هایی مانند اشاره‌گرها (pointers) و مدیریت مستقیم حافظه باعث شده است که C بتواند در سیستم‌های نیازمند بهینه‌سازی حافظه مورد استفاده قرار گیرد [1].

این زبان به دلیل ویژگی‌هایش در صنعت کاربردهای بسیار مهم دارد که در ادامه به برخی از آن‌ها اشاره می‌کنیم.

در بسیاری از سیستم‌عامل‌ها مانند Linux، Windows، و macOS بخش‌هایی از آن‌ها با استفاده از زبان C پیاده‌سازی شده است [3]. یکی از مهم‌ترین دلایل استفاده از این زبان در توسعه‌ی سیستم‌عامل‌ها، عملکرد بالا، کنترل دقیق حافظه، و امکان برنامه‌نویسی سطح پایین است. به‌علاوه، بسیاری از زبان‌های برنامه‌نویسی دیگر مانند Java، C++، و Python تحت تأثیر C توسعه یافته‌اند.

[1] Efficiency
[2] Structured Programming
[3] Performance

بسیاری از توابع اصلی پایتون (Built-in Functions) به دلیل مسائل مربوط به performance، با زبان C نوشته شده‌اند. مثلا اگر سعی کنید تابع max را خودتان در python پیاده‌سازی کنید و آن را با تابع پیش‌فرض max() در پایتون، با دادن یک لیست بزرگ به عنوان ورودی، مقایسه کنید، متوجه تفاوت performance در این دو تابع می‌شوید. با وجود اینکه time complexity هر دو تابع یکسان است، اما به دلیل این که توابع built-in در python با کمک زبان C پیاده‌سازی شده‌اند، دارای performance بسیار بهتری می‌باشند.

از این زبان در بسیاری از سیستم‌های embedded و real-time و همچنین برای ارتباط با microcontroller ها و بردهایی مانند Arduino، به دلیل سرعت و دسترسی low-level آن استفاده می‌شود. علاوه بر این، C در مواردی دیگر مانند موتورهای بازی‌سازی و توسعه‌ی کامپایلرها نیز استفاده می‌شود. در بسیاری از موتورهای بازی‌سازی مانند Unreal Engine نیز از زبان C استفاده شده است. همچنین، برخی از پایگاه‌های داده معروف مانند MySQL به زبان C نوشته شده‌اند. به دلیل سرعت بالای این زبان، از آن در پیاده‌سازی برخی از کتابخانه‌های هوش مصنوعی مانند OpenCV نیز استفاده شده است [3].

با توجه به موارد بالا، اهمیت این زبان تا حد زیادی برای ما مشخص است. سرعت بالا، قابلیت‌های دسترسی low-level و قابلیت کنترل حافظه در زبان C، در کنار انعطاف‌پذیری و کارایی بالا، آن را به یک زبان بسیار مهم برای پیاده‌سازی برنامه‌های کامپیوتری (خصوصاً برنامه‌هایی که به پردازش‌های سطح پایین و دسترسی به سخت‌افزار نیاز دارند) تبدیل کرده است.

در صورت علاقه‌مندی به یادگیری بیشتر درباره C، می‌توانید محتویات این لینک را مطالعه و بررسی کنید.

۲) ساختار کلی

در زبان برنامه‌نویسی C، کد برنامه درون یک فایل با پسوند c. قرار دارد. لازم به ذکر است که در پروژه‌های چند فایلی می‌توانیم فایل‌هایی با پسوند h. نیز تعریف کنیم. اما در این پروژه ما کدهای چند فایلی نداریم و از این موضوع صرف نظر می‌کنیم. یک برنامه به زبان **C** از قسمت‌های زیر تشکیل شده است:

- **پیش‌پردازنده‌ها (Preprocessor Directives):** شامل دستوراتی که با # شروع می‌شوند و توسط کامپایلر و پیش از کامپایل پردازش می‌شوند. ما در این پروژه از این دستورات صرف نظر می‌کنیم.
- **توابع (Functions):** در این زبان امکان تعریف تابع وجود دارد. همچنین شروع اجرا هر برنامه از تابع main است.
- **متغیرها (Variables):** برای ذخیره‌ی داده‌ها استفاده می‌شوند و باید قبل از استفاده تعریف شوند.
- **دستورات (Statements & Expressions):** عملیات‌هایی که برنامه انجام می‌دهد، مانند محاسبات، شرط‌ها، حلقه‌ها و مواردی مانند sizeof , typedef.
- **توضیحات (Comments):** بخش‌های توضیحی که توسط کامپایلر نادیده گرفته می‌شوند.

۲-۱) قواعد کلی نحو

زبان C به بزرگ و کوچک بودن حروف حساس است. در این زبان، وجود کاراکترهای tab و space تاثیری در خروجی برنامه ندارند. جزئیات مربوط به scope و خطوط برنامه در ادامه توضیح داده خواهد شد.

۲-۲) Comment ها

در این زبان کامنت‌ها را به دو صورت می‌توان نمایش داده:

- **استفاده از //** که در نتیجه از آنجا تا آخر خط به عنوان کامنت در نظر گرفته می‌شود.
- **استفاده از /* و در انتها */** که در نتیجه کل آن قسمت به عنوان کامنت در نظر گرفته می‌شود.

```
1  // this is a comment
2  int a; // this is a comment
3  /*
4      this
5      is
6      a
7      comment
8  */
```

۲-۳) قواعد نامگذاری (Identifier ها)

اسامی انتخابی برای نامگذاری باید از قواعد زیر پیروی کنند:

- تنها از کاراکترهای [a..z] ، [A..Z] ، _ و ارقام تشکیل شده باشند (محدودیتی روی تعداد کاراکترهای یک اسم در زبان C وجود ندارد).
- با رقم شروع نشوند.
- معادل کلیدواژه‌ها نباشند. در جدول زیر تمامی کلید واژه‌ها آمده است:

break	char	const	continue	double	else
float	for	if	int	long	return
short	signed	sizeof	typedef	unsigned	void
while					

- نام هر تابع یکتا است.
- نام متغیرها در هر scope یکتا است ولی در scope های درونی‌تر می‌توان از نام‌های متغیرهای scope های بیرونی استفاده کرد. در این حالت، در طول آن scope متغیر تعریف شده در scope درونی هنگام استفاده ارجحیت دارد. اصطلاحاً از Variable shadowing (که در ادامه‌ی درس می‌خوانید) پشتیبانی می‌شود.

۲-۴)typedef

در زبان برنامه نویسی C، کلمه کلیدی به نام typedef وجود دارد که به منظور تعریف یک نام جدید برای انواع داده‌ی موجود استفاده می‌شود. در مثال زیر عبارت BYTE برای نوع داده‌ی unsigned char تعریف شده است:

```
1  typedef unsigned char BYTE;
```

بعد از تعریف بالا، زمانی که شما از عبارت BYTE استفاده می‌کنید، کامپایلر آن را معادل unsigned char در نظر می‌گیرد. در حقیقت یک نام مستعار برای یک نوع خاص تعریف می‌شود. برای نمونه:

```
BYTE b1, b2;
```

بر اساس قرارداد از حروف بزرگ برای نام جدید استفاده می‌شود تا به برنامه‌نویس یاد آوری کند که این نوع داده را خود آن برنامه‌نویس یا برنامه‌نویس دیگر تعریف کرده است و یک نام مستعار می‌باشد. اما می‌توان از حروف کوچک هم مانند نمونه‌ی زیر استفاده کرد:

```
typedef unsigned char byte;
```

۳) انواع داده

۳-۱) انواع type های پایه

تعریف متغیر در زبان C، نام و ویژگی‌های مختلفی از یک متغیر را مشخص می‌کند. یکی از این ویژگی‌ها type آن متغیر است. در این زبان می‌توان از چهار type پایه استفاده کرد که به صورت زیر می‌باشند.

void	char	int	float	double

۳-۲) انواع type specifier

در این زبان با استفاده از type specifier های گوناگون می‌توان تغییراتی در متغیر مورد نظر ایجاد کرد. از ویژگی‌هایی که در زمان تعریف یک متغیر می‌توان مشخص کرد، مقدار حافظه مورد نیاز برای آن متغیر و روش ذخیره‌سازی آن در حافظه است. از کنار هم گذاشتن ویژگی‌های متفاوت نیز می‌توان ویژگی‌های جدیدی تولید کرد. برای مثال، با استفاده از کلیدواژه long، می‌توان عددی با سایز 32 بیت تولید کرد. سپس با اضافه کردن کلیدواژه unsigned می‌توان مشخص کرد که عدد به صورت بدون

علامت در حافظه ذخیره شود و مورد استفاده قرار گیرد. از ترکیب این ویژگی‌ها نیز می‌توان بهره برد.
به عنوان نمونه، چند مثال از تعریف انواع داده در جدول زیر نمایش داده شده‌اند:

Type	Size (bits)	Size (bytes)	Range
char	8	1	-128 to 127
unsigned char	8	1	0 to 255
int	16	2	-2^{15} to $2^{15}-1$
unsigned int	16	2	0 to $2^{16}-1$
short int	8	1	-128 to 127
unsigned short int	8	1	0 to 255
long int	32	4	-2^{31} to $2^{31}-1$
unsigned long int	32	4	0 to $2^{32}-1$
float	32	4	3.4E-38 to 3.4E+38
double	64	8	1.7E-308 to 1.7E+308
long double	80	10	3.4E-4932 to 1.1E+4932

[4]

۳-۳) توصیف چند متغیر در یک خط

زبان C از امکان توصیف چند متغیر از یک نوع در یک خط پشتیبانی می‌کند. به این صورت که می‌توان بعد از مشخص کردن ویژگی‌های مورد نظر، نام متغیرهای مورد نیاز را با یک ',' در میانشان مشخص کرد.
در تصویر زیر مثالی از این نوع تعریف متغیرها آمده است.

```
1  int a, b = 1, c;
```

۴-۳) تعریف متغیرهایی از نوع pointer

علاوه بر type های گفته شده، می‌توان متغیرهایی از نوع اشاره‌گر نیز تعریف کرد، به گونه‌ای که به خانه‌ای از حافظه اشاره کنند. این نوع متغیرها هنگام declaration با استفاده از ' * ' بعد از type توصیف می‌شوند.

[4] https://www.startertutorials.com/blog/data-types-c.html

```
1  char* c;
```

۴) عملگرها

عملگرها دقیقا مشابه زبان C اصلی است. تنها تفاوت در عدم وجود عملگرهای '.' و '->' است. این دو عملگر برای دسترسی به اطلاعات درون ساختمان‌ها [5] مورد استفاده قرار می‌گرفتند. از آن جا که در این پروژه، ساختمان‌ها را پیاده سازی نمی‌کنیم به این دو عملگر نیازی نخواهیم داشت؛ بنابراین این دو در زبان ما وجود ندارند.

۴-۱) عملگرهای حافظه

در زبان C برای مدیریت و دسترسی به حافظه عملگرهایی تعریف شده‌اند. برای دستیابی به آدرس حافظه یک متغیر می‌توان از عملگر '&' استفاده کرد و برای دسترسی به محتویات یک آدرس حافظه می‌توان از عملگر '*' استفاده کرد. لیستی از عملگرهای حافظه در جدول زیر آمده است.

عملگر	شرکت‌پذیری	توضیح	مثال
*	راست	دسترسی به محتوای حافظه به کمک آدرس	*A
&	راست	دسترسی به آدرس حافظه	&A

۴-۲) عملگرهای حسابی

این دسته از عملگرها تنها روی اعداد اعمال می‌شوند. لیست این عملگرها در جدول زیر آمده است. در مثال‌ها A برابر با 20 و B برابر با 10 هستند.

[5] Struct

عملگر	شرکت‌پذیری	توضیح	مثال
+	چپ	جمع	A+B=30
-	چپ	تفریق	A–B=10
*	چپ	ضرب	A*B=200
/	چپ	تقسیم	A/ B = 2 B / A =0
%	چپ	باقی‌مانده	A % B=0
-	راست	منفی تک عمل‌وندی	-A = -20
-- و ++	راست	پیشوندی	--A
-- و ++	چپ	پسوندی	A++

۳-۴) عملگرهای مقایسه‌ای

این عملگرها وظیفه‌ی مقایسه را دارند؛ توجه داشته باشید که عملوندهای عملگرهای > و < تنها از جنس اعداد هستند. همچنین برای عملگر == و != نیز باید تایپ عملوندها با یکدیگر همخوانی داشته باشند. لیست عملگرهای مقایسه‌ای در جدول زیر آمده است.

عملگر	شرکت‌پذیری	توضیح	مثال
==	چپ	تساوی	(A==B)
!=	چپ	عدم تساوی	(A!=B)

<	چپ	کوچکتر	(A<B)
>	چپ	بزرگتر	(A>B)
<=	چپ	کوچکتر یا مساوی	(A<=B)
>=	چپ	بزرگتر یا مساوی	(A>=B)

۴-۴) عملگرهای بیتی

در زبان C، تعدادی عملگر بیتی نیز تعریف شده‌اند که برای برنامه‌نویسان امکان اعمال تغییرات در سطح بیت را فراهم می‌کنند. لیستی از انواع این عملگرها در جدول زیر آمده است. در مثال‌ها A و B برابر با 1، C برابر با 8 و D برابر با 2 هستند.

عملگر	شرکت‌پذیری	توضیح	مثال
&	چپ	عطف	(A & B) = 1
|	چپ	فصل	(A | B) = 1
^	چپ	XOR	(A ^ B) = 0
>>	چپ	شیفت بیتی به راست	(C>>D) = 2
<<	چپ	شیفت بیتی به چپ	(C << D) = 32
~	راست	نقیض بیتی	(~A) = 0

۴-۵) عملگرهای منطقی

در زبان C، متغیری از جنس boolean نداریم و با استفاده از اعداد، حاصل عبارات منطقی را مدل‌سازی می‌کنیم. به این صورت که هر مقداری غیر از 0، true در نظر گرفته می‌شود و خود عدد 0، false در نظر گرفته می‌شوند. همچنین در این زبان می‌توان با استفاده از عملگرهای منطقی، عبارات منطقی پیچیده‌تر را بیان کرد. لیست این نوع عملگرها در جدول زیر آمده است. در مثال‌ها A برابر با 1 و B برابر با 0 هستند.

عملگر	شرکت‌پذیری	توضیح	مثال
&&	چپ	عطف منطقی	(A && B) = 0
||	چپ	فصل منطقی	(A || B) = 1
!	راست	نقیض منطقی	(!A) = 0

۴-۶) عملگر تخصیص

این عملگر که به صورت = نمایش داده می‌شود وظیفه‌ی تخصیص را برعهده دارد. یعنی مقدار عملوند سمت راست را به عملوند سمت چپ اختصاص می‌دهد. همچنین دقت داشته باشید که عملوند سمت چپ باید از نوع left-value باشد. عبارات left-value عباراتی هستند که به یک مکان در حافظه اشاره می‌کنند. عبارات right-value لزومی ندارد به مکان خاصی در حافظه اشاره کنند و صرفا کافی است یک عبارت دارای مقدار باشند. به عنوان مثال یک متغیر یا یک دسترسی به یکی از عناصر آرایه یک عبارت left-value است اما عبارت 3 + 10 یک عبارت right-value محسوب می‌شود. عبارات right-value تنها در سمت راست عملگر تخصیص قرار می‌گیرند.

۷-۴) عملگر شرطی

این عملگر برای راحتی و کم کردن حجم کد مورد استفاده قرار می‌گیرد. به طور کلی این عملگر یک شرط را بررسی می‌کند. اگر شرط برقرار بود مقدار اول و در غیر این صورت مقدار دوم مشخص شده را می‌گیرد. بنابراین این عملگر از سه قسمت اصلی تشکیل شده است؛ شرط عملگر، مقدار در صورت برقرار بودن آن و در نهایت مقدار در صورت برقرار نبودن آن. نکته‌ی حائز اهمیت در مورد این عملگر استفاده از آن به صورت بازگشتی است. در تصویر زیر مثالی از استفاده از این عملگر مشخص است.

```
var = condition ? value_if_true : value_if_false;
// Example
var = (x > 0) ? 1 : -1;
```

۷-۴) اولویت عملگرها

اولویت عملگرها طبق جدول زیر است:

شرکت‌پذیری	عملگرها	دسته	اولویت
چپ به راست	( ), [ ]	parenthesis	۱
راست به چپ	-, --, ++, size_of(), &, *, ~	Unary operators	۲
چپ به راست	*, /, %	Arithmetic multiply, divide and remainder	۳
چپ به راست	+, -	Arithmetic add and subtract	۴
چپ به راست	>>, <<	Bitwise shift	۵
چپ به راست	<, <=, >=, >	Relational	۶

۷	Equality	=!, ==	چپ به راست
۸	Bitwise AND	&	چپ به راست
۹	Bitwise XOR	^	چپ به راست
۱۰	Bitwise OR	|	چپ به راست
۱۱	Logical AND	&&	چپ به راست
۱۲	Logical OR	||	چپ به راست
۱۳	Conditional	?	راست به چپ
۱۴	Assignment	=, *=, /=, %=, +=, -=, <<=, >>=, &=, |=, ^=	راست به چپ
۱۵	comma	,	چپ به راست

۵) گزاره‌های شرطی

گزاره‌های شرطی برای تصمیم‌گیری استفاده می‌شوند. در زبان C ساختار شرطی به حالت زیر خواهد بود: هر ساختار شرطی با یک if...then آغاز می‌شود. هر شرط می‌تواند else داشته باشد یا نداشته باشد. همچنین هر ساختار شرطی می‌تواند else if داشته باشد یا نداشته باشد (else if در صورت وجود می‌تواند چند بار استفاده شود اما else در صورت وجود فقط یک بار می‌آید).
قطعه‌کد زیر مثالی از این ساختار را نشان می‌دهد.

```c
if (x > 3) {
    printf("x is greater than 3\n");
} else if (x == 3) {
    printf("x is equal to 3\n");
} else {
    printf("x is less than 3\n");
}
```

۶) حلقه‌ها (Loops)

حلقه‌ها به منظور تکرار یک بلوک کد استفاده می‌شوند تا زمانی که شرط خاصی را برآورده کنند. در زبان C، دو نوع ساختار تکرار وجود دارد:

۶-۱) for Loop

ساختار حلقه for به شرح زیر است:

```c
for (initialization; condition; update) {
    // loop body
}
```

Initialization: این بخش در ابتدا اجرا می‌شود و برای تعریف متغیر کنترلی حلقه استفاده می‌شود. در این قسمت یا متغیری تعریف می‌شود یا متغیری که از قبل تعریف شده مقدار اولیه می‌گیرد.

Condition: پیش از هر بار اجرای بدنه‌ی حلقه، این شرط بررسی می‌شود اگر شرط درست بود حلقه اجرا می‌شود، و در غیر این صورت از حلقه بیرون می‌آید. این شرط باید از نوع boolean باشد.

Update: این قسمت پس از هر بار اجرای بدنه‌ی حلقه انجام می‌شود و متغیر کنترلی حلقه را به‌روزرسانی می‌کند.

قطعه‌کد زیر مثالی از استفاده حلقه for است:

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

while Loop (۲-۶

ساختار حلقه while به شکل زیر است:

```c
while (cond) {
    // Body
}
```

Condition: این حلقه تا زمانی که شرط تعیین‌شده برقرار باشد، تکرار می‌شود. شرط این حلقه باید از نوع boolean باشد و زمانی که شرط برقرار نباشد، حلقه متوقف می‌شود.

در هر دو نوع حلقه می‌توان از دستورات break و continue برای کنترل جریان استفاده کرد.

Break: دستور break، بلافاصله حلقه را خاتمه می‌دهد و برنامه به بعد از حلقه منتقل می شود.

Continue: دستور continue، اجرای تکرار[6] فعلی حلقه را متوقف می‌کند و با تکرار بعدی ادامه می یابد.

کد زیر مثالی از استفاده حلقه while است:

```c
while (x < 5) {
    foo(x);
}
```

[6] Iteration

۷) منابع

[1] Kernighan, B.W. and Ritchie, D.M., 1988. *The C programming language*. prentice-Hall.

[2] Ritchie, D.M., 1993. The development of the C language. *ACM Sigplan Notices*, *28*(3), pp.201-208.

[3] Programiz. (2023) *8 Main Uses of C Programming in 2023*. Available at: https://programiz.pro/resources/c-uses/