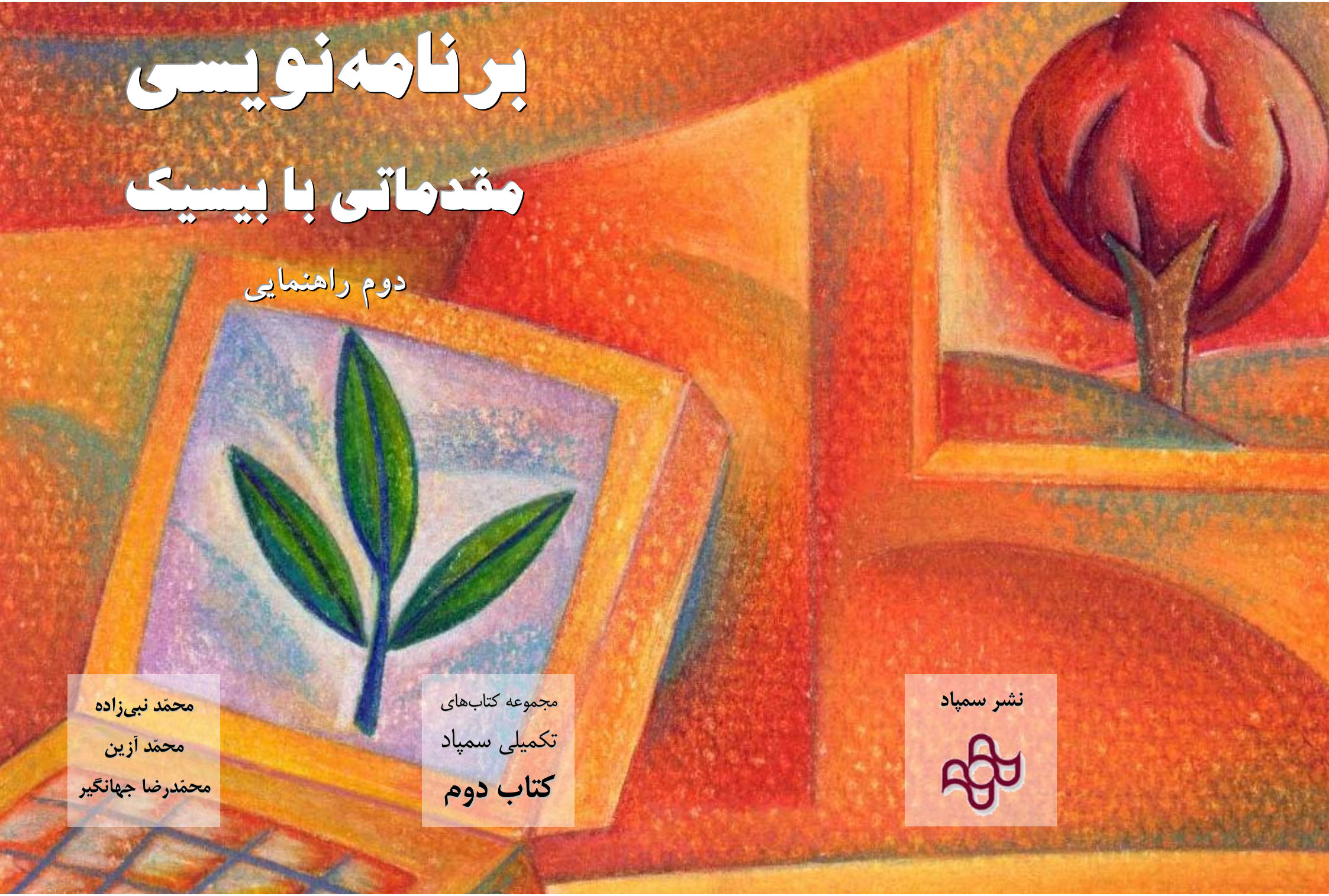
برنامه‌نویسی مقدماتی با بیسیک

دوم راهنمایی



محمّد نبی‌زاده

محمّد آزین

محمّدرضا جهانگیر

مجموعه کتاب‌های

تکمیلی سمپاد

کتاب دوم

نشر سمپاد

به نام پروردگار

برنامه‌نویسی مقدماتی با بیسیک

دوم راهنمایی

محمّد نبی‌زاده

محمّد آزین

محمّدرضا جهانگیر

مجموعه کتاب‌های تکمیلی سمپاد

کتاب دوم

نشر سمپاد

نشر سمپاد

برنامه‌نویسی مقدماتی با بیسیک

پایهٔ اول راهنمایی

مجموعه کتاب‌های تکمیلی سمپاد، کتاب اول

نویسندگان: محمد نبی‌زاده، محمد آزین، محمدرضا جهانگیر

طرح جلد، صفحه آرایی و تصویرسازی: محمد سیاحت‌گر

حروف چینی: نشر سمپاد

شمارگان : ۱۰/۰۰۰ جلد

چاپ اول: ۱۳۸۶

چاپ و صحافی:طیف نگار

قیمت : ۱۰۰۰ تومان

شابک

ISBN

فهرست مطالب

فهرست مطالب

مقدمه

امروزه رایانه برای اکثر دانشمندان و متخصصین، نقش یک قلم را پیدا کرده است. این افراد با جدیت و پشتکار و با تسلطی که از پیش بر علوم پایه‌ی رایانه به دست آورده‌اند، همگام با دانش روز دنیا، در مسیر تکامل علمی خویش گام برمی‌دارند. از سوی بسیاری از دانشمندان علوم رایانه، تکنولوژی آموزشی و حتی متخصصین روان‌شناسی، از برنامه‌نویسی و برنامه‌سازی در بین علوم پایه‌ی رایانه، به عنوان یکی از ابزارهای مفید و کلیدی، برای آموزش اصول تفکر منطقی و منظم و هم‌چنین توسعه‌ی خلاقیت‌های فردی و تحصیلی یاد می‌شود.

یک برنامه‌نویس با دید وسیع‌تر و بهتری به وقایع و مسایل اطراف خود می‌نگرد. او مجهز به مهارت‌ها و فنونی است که به او یاد می‌دهند که چگونه از مسایل بزرگ و پیچیده نهراسد، چگونه با دید منطقی و منظم خود، مسایل بزرگ را به مسایل کوچک‌تر بشکاند و سپس نسبت به حل این مسایل کوچک‌تر و ساده‌تر اقدام نماید. او به خوبی یاد گرفته است که چگونه از رایانه به عنوان یک قلم استفاده کند.

یک برنامه نویس لزوماً متخصص رایانه نیست. حافظه‌ی ما، بسیاری از دانش‌آموختگان قدیمی و جدید سمپاد را به یاد دارد، که اگر چه در رشته‌ها و گرایش‌های علوم رایانه تحصیل نکرده‌اند، اما به واسطه‌ی تسلط بر برنامه‌نویسی، همواره از دیگر هم‌کلاسی‌ها و هم‌دوره‌ای‌های خود، چه در مراکز علمی و آکادمیک و چه در محیط‌های کاری، پیش بوده و هستند؛ چرا که برنامه‌نویسی به آنها آموخته بود، هیچ‌گاه و در هیچ موقعیتی، از برخورد با مسایل ناشناخته و بزرگ، ترسی نداشته باشند و خلاقیت و تجارب قبلی خود را در حل مسایل جدید، به کار ببندند.

تیم نویسنده‌ی این کتاب، حاصل تجربیات طولانی و گران‌سنگ خود و اساتیدشان، در گروه‌های کامپیوتر مراکز راهنمایی و دبیرستان علامه‌حلی تهران، را در قالب این کتاب جمع‌آوری و تدوین نموده‌اند. کتاب حاضر حاصل گردآوری تلاش‌ها و تجربیات بیش از ۱۶ سال تدریس درس برنامه‌نویسی در طی جلسات مختلف گروهی، برای دانش‌آموزانی است که هم‌اکنون در اقصی نقاط دنیا، مهارت‌ها و فنون اکتسابی خود را در شاخه‌های مختلف علم و فناوری به کار می‌بندند.

نویسندگان کتاب وظیفه‌ی خود می‌دانند از همه‌ی پیش‌کسوتان و اساتید خود در گروه کامپیوتر مرکز علامه‌حلی تهران، به ویژه آقایان رضا صدیق، فربد رزازی و سعید سرکاراتی و دیگر همکاران محترم گروه از سال ۱۳۷۰ تا کنون تشکر و قدردانی نماید. همچنین از آقای محمّد سیاحت‌گر که وظیفه‌ی ویراستاری و سرکار خانم ریحانه کبیری که زحمت تایپ مطالب کتاب را بر عهده داشتند، تشکر و قدردانی می‌گردد.

در خاتمه جا دارد یاد شهدای سمپاد، معلمین و اساتید خود، بویژه آن‌هایی که اعتقاد داشتند – تیزهوشان کسانی هستند که توان‌مندی بیشتری برای خدمت به خلق خدا دارند – را گرامی بدارم.

به امید توفیق روز افزون در سایه‌ی الطاف الهی

محمدرضا جهانگیر

دانش‌آموخته و معلم رایانه‌ی سمپاد

سخنی با دانش‌آموز

همیشه به یاد داشته باشید که یادگیری علوم رایانه، مانند یادگیری فنون شنا است. آیا شما تا به حال سراغ دارید، که کسی بدون تمرین در استخر و تنها با خرید یک کتاب آموزش شنا، توانسته باشد شناگر ماهری شود ؟!

به همین خاطر، خیلی جدی به شما توصیه می‌کنیم که صرفا به حل مثال‌ها و تمرین‌های این کتاب بر روی کاغذ، اکتفا نکنید و حتما برنامه‌های خود را بر روی رایانه اجرا کنید.

کسب مهارت‌ها و شگردهای برنامه‌نویسی، بدون سر و کله زدن با رایانه و گرفتن جواب دلخواه‌تان از رایانه، ممکن نیست. اگر بخواهیم خیلی راحت و بی‌پرده سخن بگوییم، یک برنامه‌نویس خوب دو ویژگی مهم دارد: در درجه‌ی اول از هوش و استعداد خوب و در درجه‌ی دوم از صبر و تحمل بالایی بهره‌مند است؛ چرا که جواب گرفتن از یک مجری غیر خلاق مانند رایانه، احتیاج به صبر و تحمل قابل توجهی دارد و لذا یادگیری برنامه-نویسی با عجله و مثلا در ظرف یک هفته و یا چند روز، به طور فشرده و تضمینی!، اصلا عاقلانه و شدنی نیست.

پس از مایی که سالیان سال است این راه را رفته‌ایم، به شما نوگلِ نوپا در این عرصه، چند نصیحت :

حتما مثال‌ها و تمرین‌های کتاب را بر روی رایانه اجرا کنید.

سعی کنید قسمت‌هایی که با عنوان «برای مطالعه‌ی بیشتر و یا تمرینات ویژه‌ی دانش‌آموزان علاقه‌مند» مشخص شده است را در ساعات خارج از کلاس دنبال کنید.

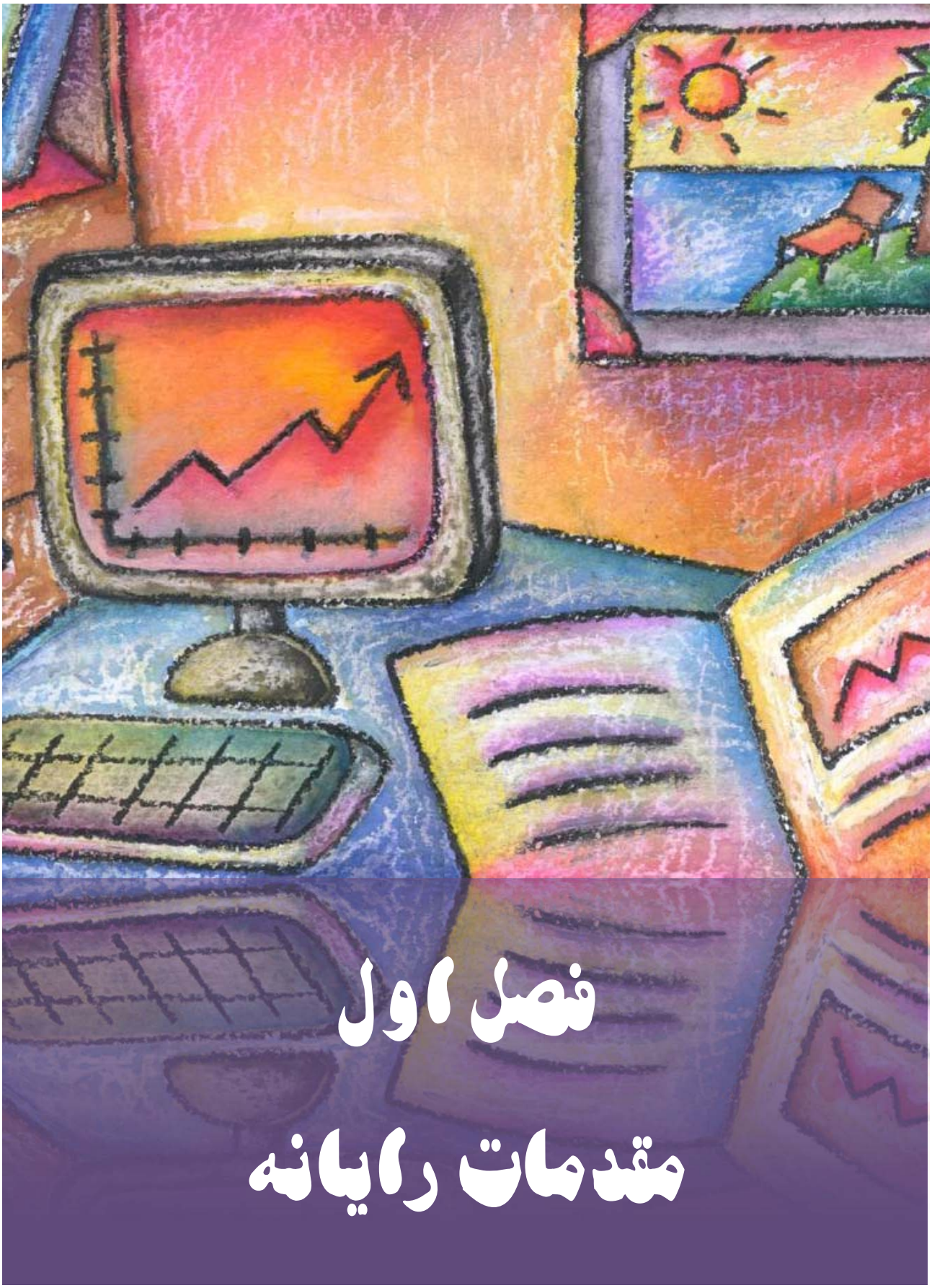
بهترین تمرین‌ها، تمرین‌هایی هستند که خودتان طرح می‌کنید. سعی کنید به همراه معلم و دوستان خود، تمرین‌ها و مسایل جدیدی را برای برنامه‌نویسی پیشنهاد کرده و آن‌ها را پای رایانه حل کنید.

برنامه‌نویسی رایانه، درس شیرینی است که به مرور زمان اثرات مثبت خود را بر ذهن، فکر و حتی برنامه‌ی زندگی شما به جا می‌گذارد. نقش این درس در توسعه‌ی خلاقیت‌ها و آموزش تفکر منطقی و منظم، امروزه بر همه ثابت شده است.

حتی شما دانش‌آموز عزیز، که نمی‌خواهید در دوره‌ی دبیرستان رشته‌ی ریاضی فیزیک را انتخاب کنید، بدان که استفاده‌ی شما از برنامه‌نویسی در سایر دروس، کمک شایانی به پیشرفت شما در دروس مورد علاقه‌یتان می‌کند. پس قدر این درس را بیشتر بدانید.

موفق باشید!

تیم نویسنده‌ی کتاب



فصل اول

مقدمات رایانه

۱. ۱. مقدمه

امروزه رایانه به صورت یک ابزار جدایی ناپذیر از زندگی انسان‌ها درآمده است که استفاده‌ی درست از آن نیازمند شناخت دقیق ساختار، توانایی‌ها و نحوه‌ی تعامل با رایانه می‌باشد. در درس رایانه‌ی سال گذشته با ساختار رایانه و نحوه‌ی کار کردن با برخی از مهمترین نرم‌افزارهای کاربردی آن آشنا شدید. امسال پس از یک مرور سریع، با جنبه‌ی دیگری از توانایی‌های رایانه یعنی *قابلیت برنامه ریزی*، آشنا می‌شویم.

۱. ۲. ساختار رایانه

همانطور که می‌دانید به هر ماشین محاسبه‌گر، کامپیوتر یا رایانه گفته می‌شود. صرف نظر از ماشین‌های محاسبه‌ی مکانیکی که امروزه استفاده‌ی عمومی ندارند، می‌توانیم هر سیستم محاسبه‌گری که قابل برنامه‌ریزی باشد را رایانه بنامیم. این سیستم‌ها می‌توانند طیف گسترده‌ای از ماشین‌های لباس‌شویی تا ابررایانه‌های پیش‌بینی کننده‌ی وضع هوا را در بر بگیرند.

هر رایانه از یک *واحد پردازشگر مرکزی* (یا CPU) برای اجرای دستورات و انجام محاسبات استفاده می‌کند. در کنار این واحد، ابزارهای *ورودی*، *خروجی* و *واحد حافظه* قرار دارند. داده‌ها از جهان خارج به وسیله‌ی ورودی‌ها وارد رایانه شده و پاسخِ محاسبه شده توسط خروجی‌ها به کاربر ارائه می‌شود. واحد حافظه به ذخیره‌ی نتایج و اطلاعات مورد نیاز برای انجام محاسبات می‌پردازد. مطمئناً بدون وجود حافظه‌ای که اطلاعات اولیه را در اختیار داشته باشد و پاسخ‌های نهایی را در خود ذخیره کند، محاسبات بی معنی و بی استفاده خواهند بود.

در رایانه سه نوع حافظه وجود دارد:

۱. حافظه‌ی پنهان
۲. حافظه‌ی موقت
۳. حافظه‌ی دائم

برای روشن شدن جایگاه هر یک از این سه نوع حافظه، از یک مثال استفاده می‌کنیم و بعد به شرح هر یک از این انواع می‌پردازیم. فرض کنید دانش‌آموز تیزهوشی در آزمون ریاضی شرکت کرده است. این دانش‌آموز برای پاسخ به آزمون، مسیر زیر را طی می‌کند :

ابتدا با استفاده از ابزار ورودی یعنی چشم، صورت مسئله را مطالعه می‌کند. سپس داده‌های مسئله در حافظه‌ی مغز قرار می‌گیرند و مغز شروع به محاسبه می‌کند. تا جایی که محاسبات کوتاه هستند، مغز می‌تواند از حافظه‌ی کوتاه مدت استفاده کرده و جواب این محاسبات را به دست آورد. حافظه‌ی کوتاه مدت مغز شبیه حافظه‌ی پنهان پردازنده است. برای انجام محاسبات پیچیده‌تر، مغز به یک حافظه‌ی

کمکی نیازمند است، این حافظه همان صفحه‌ی چرک‌نویس است که گنجایش بیشتری برای ذخیره‌سازی محاسبات دارد. این حافظه معادل حافظه‌ی موقت در رایانه می‌باشد. دانش‌آموز در پایان، پاسخ‌هایی را که مغز آن‌ها را محاسبه کرده و در برگه‌ی چرک‌نویس ذخیره شده‌اند را به برگه‌ی پاسخ‌نامه منتقل می‌کند؛ برگه‌ی پاسخ‌نامه مثالی از حافظه‌ی دائم می‌باشد.

با توجه به مثال فوق، تعریف دقیق‌تری از این سه نوع حافظه ارائه می‌دهیم :

۱. *حافظه‌ی پنهانی* : این حافظه گنجایش بسیار کم و سرعت انتقال اطلاعات بسیار بالایی دارد، بنابرین برای انجام محاسبات سریع و کوچک استفاده می‌شود.
۲. *حافظه‌ی موقت* : اطلاعات مورد نیاز برای حل یک مسئله و یا محاسبات لازم برای حل یک مسئله بر روی این حافظه ذخیره می‌شوند. از ویژگی‌های این حافظه، سریع بودن آن و پاک شدن اطلاعات آن در اثر قطع جریان برق می‌باشد.
۳. *حافظه‌ی دائم* : برنامه‌ها و نتایج محاسبات در نهایت بر روی این حافظه ذخیره می‌شوند. می‌توان اطلاعات را بر روی این حافظه به طور دائم نگهداری کرد، لذا محتوای این حافظه در اثر قطع جریان برق پاک نمی‌شود.

۱. ۳. چگونه با رایانه ارتباط برقرار کنیم؟

حتماً شنیده‌اید که زبان رایانه صفر و یک است. زبان عجیبی است! نه؟ در مدارهای الکترونیکی رایانه، محاسبات تنها با ارقام صفر و یک انجام می‌شوند، به این روش محاسبه، محاسبه در مبنای ۲ می‌گوییم. ما انسان‌ها محاسباتمان را در مبنای ۱۰ انجام می‌دهیم (چرا؟) به همین خاطر است که از ارقام ۰، ۱، ۲، ۳، ... ، ۸ و ۹ استفاده می‌کنیم؛ حتماً رایانه هم از این روش تعجب خواهد کرد ! (شما روش محاسبه در مبناهای مختلف را امسال در درس ریاضی خواهید آموخت.)

در واقع رایانه تمام اعداد و دستورات را به صورت دنباله‌هایی از صفر و یک می‌فهمد. بنابراین برای ارتباط برقرار کردن با آن نیز باید به همین روش عمل کرد، یعنی باید با زبان صفر و یک با رایانه

صحبت کرد ! این کار بسیار مشکل است و عملاً انجام کارهای کمی پیچیده با این روش غیر ممکن خواهد بود. اما این مشکل راه حلی دارد : *زبان‌های برنامه نویسی*.

. . . . . .

۱. ۴. زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی، برقراری ارتباط ما با رایانه را راحت‌تر می‌کنند. این زبان‌ها ساختاری شبیه به زبان انسان دارند تا کاربر بتواند به راحتی منظور خود را بیان کند، سپس این زبان، به زبان صفر و یک *ترجمه* می‌شود تا برای رایانه قابل فهم باشد. به این روش می‌توانیم بسیار راحت از رایانه بخواهیم که کارهای مورد نظر ما را انجام دهد.

طبق آنچه گفتیم زبان‌های برنامه‌نویسی در سطوح مختلفی طبقه‌بندی می‌شوند، به این ترتیب که هر چه زبان برنامه‌نویسی به زبان انسان نزدیک‌تر باشد به آن زبان، *سطح بالاتر* می‌گوییم و هر چه به زبان رایانه (صفر و یک) نزدیکتر باشد، آن زبان را *سطح پایین‌تر* می‌گوییم.

بر اساس این تقسیم بندی، نگاهی به معروف‌ترین زبان‌های برنامه‌نویسی موجود (از سطح پایین به بالا) می‌اندازیم :

- **زبان اسمبلی (Assembly) :** این زبان نزدیک ترین زبان به زبان صفر و یک رایانه است. در واقع یک رابطه‌ی ساده‌ی یک به یک بین دستورات صفر و یکِ رایانه و دستورات زبان اسمبلی وجود دارد. برنامه نویسی با زبان اسمبلی، هر چند از کارکردن با صفر و یک‌ها ساده‌تر است، ولی هنوز فاصله‌ی زیادی با زبان انسان دارد. از طرف دیگر زبان اسمبلی زبان بسیار قدرتمندی است، زیرا به طور مستقیم با پردازنده‌ی رایانه در ارتباط است و پردازنده را کاملاً در کنترل خود می‌گیرد.

```
.model small
.stack
.data
message  db "Hello World!",
"$"

.code

main   proc
  mov   ax,seg message
  mov   ds,ax

  mov   ah,09
  lea   dx,message
  int   21h

  mov   ax,4c00h
  int   21h
main   endp
end main
```

برنامه‌ای برای نوشتن یک پیغام ساده در زبان اسمبلی !

- **زبان‌های C و C++ :** زبان C ، زبان سطح بالاتر از زبان اسمبلی است، پس می‌توانید حدس بزنید که برنامه‌نویسی با آن باید ساده‌تر از اسمبلی باشد. هم‌چنین این زبان، زبان قدرتمندی نیز هست. خوب است بدانید که زبان C را با استفاده از زبان اسمبلی ساخته‌اند. نسخهٔ جدیدتر زبان C به نام C++ شناخته می‌شود.

```
#include <iostream.h>

int main()
{
      cout << "Hello World!" << endl;
      return 0;
}
```

همان برنامه در زبان C++

- **زبان پاسکال :** دستورات زبان پاسکال بسیار شبیه به کلمات زبان انگلیسی هستند، بنابرین یادگیری و برنامه‌نویسی با آن، راحت‌تر از زبان C است و در نتیجه این زبان از زبان C سطح بالاتر است.

```
program HelloWorld(output);
begin
    WriteLn('Hello World!');
end.
```

باز هم همان برنامه در زبان پاسکال

- **زبان بیسیک (BASIC) :** زبان بیسیک یکی دیگر از زبان‌های سطح بالا است. یادگیری زبان بیسیک و برنامه‌نویسی با آن بسیار ساده است. ما نیز به همین دلیل آن را برای آموزش در این کتاب انتخاب کرده‌ایم. شما نیز به زودی لذت برنامه‌نویسی در بیسیک را حس خواهید کرد ! زبان بیسیک نسخه‌های مختلفی دارد که ما معروف‌ترین آن‌ها یعنی Quick BASIC (بیسیک سریع!) را انتخاب کرده‌ایم.

```
PRINT “Hello World!”
```

و در آخر همان برنامه به زبان بیسیک.

آیا می‌توانید حدس بزنید که این برنامه چه کاری انجام می دهد ؟!

- **زبان‌های بصری (Visual) :** زبان‌های بصری مانند Visual Basic و Visual C++ قابلیت‌های جدیدی را به زبان‌های پیشین اضافه کرده‌اند. شما می‌توانید در این زبان‌ها علاوه بر نوشتن برنامه، از امکانات آماده‌ای که به صورت تصویری در محیط برنامه‌نویسی گنجانده شده‌اند، استفاده کنید. این کار برنامه‌نویسی در محیط سیستم‌عامل *ویندوز* را بسیار راحت‌تر می‌کند. سال آینده شما با زبان Visual Basic آشنا خواهید شد.

- **سایر زبان‌ها :** در دنیای برنامه‌نویسی از زبان‌های برنامه‌نویسی بسیاری استفاده می‌شود که هر کدام ویژگی‌ها و توانایی‌های مخصوص به خود را دارند. از جملهٔ آن‌ها می توان به زبان‌های جاوا، C# ، دلفی، پرل، فرترن، PHP و ... اشاره کرد.

۱. ۵. الگوریتم

شما در خانه نشسته‌اید و مشغول درس خواندن هستید. دوستتان با شما تماس می‌گیرد و مشکلی را با شما در میان می‌گذارد. مهمانی سرزده برای او از راه رسیده است؛ ولی او نمی‌داند چگونه باید چای درست کند ! دوستتان از شما درخواست کمک می‌کند، چگونه به او کمک خواهید کرد؟

حتماً روش انجام کار را مرحله به مرحله برای او شرح خواهید داد. مثلاً خواهید گفت : اول آب را در قوری بریز، بعد قوری را روی شعله قرار بده تا آب به جوش آید، سپس یک قاشق چای خوری چای خشک در قوری بریز و شعله را کم کن، در نهایت پنج دقیقه صبر کن تا چای دم بکشد.

پرسش: می‌توانید بگویید این دستورالعمل باید چه ویژگی‌هایی داشته باشد؟

پاسخ : هر دستورالعملی باید دارای این ویژگی‌ها باشد:

۱. زبان قابل فهم برای اجرا کننده داشته باشد.
۲. مراحل باید تا حدی ساده باشند که اجرا کننده بتواند تک‌تک آن‌ها را اجرا کند.
۳. مراحل باید ترتیب صحیح داشته باشند.

به روش انجام کار که طبق قوانین بالا نوشته شده باشد، *الگوریتم* می‌گویند. *محمّد ابن موسی خوارزمی* دانشمند بزرگ ایرانی، اولین کسی است که این راه را برای بیان روش انجام کارها پیشنهاد داد. لفظ الگوریتم نیز از نام الخوارزمی گرفته شده است.

در این کتاب یاد می‌گیریم که چگونه راه حل مسئله را در قالب الگوریتم بیان کنیم و چگونه این الگوریتم را به برنامه‌ی رایانه‌ای تبدیل کنیم.

۱. ۶. برنامه

رایانه دستگاهی است که توانایی اجرای برنامه‌های مختلف را دارد. اگر به این مسئله توجه داشته باشیم که رایانه یک ابزار است وماننند سایر ابزارها وظیفه‌ی آسان‌تر کردن کارهای ما را دارد، متوجه می‌شویم که برنامه‌ها باید کاربرد خاصی برای ما داشته باشند. مثلاً بتوانند تمرینات ریاضی ما را حل کنند(!) یا اطلاعات کارکنان یک شرکت را به سرعت و راحتی در اختیار ما بگذارند و یا به پیش‌بینی وضعیت هوا در ساعات آینده بپردازند. شما می‌توانید مثال‌های گوناگونی را به این مجموعه اضافه کنید. در تمام این مثال‌ها برنامه، اطلاعاتی را از کاربر می‌گیرد که به آن *ورودی* می‌گوییم، سپس این اطلاعات را *پردازش* می‌کند تا به جواب خواسته شده‌ی کاربر برسد و در نهایت پاسخ به عنوان *خروجی* برنامه به کاربر ارائه می‌شود. (ورودی، خروجی، پردازش)

یک برنامه از دستوراتی شکل گرفته که پشت سر هم اجرا می‌شوند. این دستورات، در هر زبان برنامه‌نویسی شکل و کارکرد خاصی دارند. از پشت سر هم قرار گرفتن دستورات یک برنامه و اجرای آن توسط رایانه، کار خواسته شده برای ما انجام می‌شود. کسی که می‌خواهد با زبان خاصی برنامه بنویسد، باید ابتدا دستورات مورد نیازش را بشناسد و شکل و کاربرد آن‌ها را بداند. در مرحله بعد برنامه نویس، با پشت سر هم قرار دادن این دستورات، الگوریتم مورد نظرش را به برنامه تبدیل می‌کند.

```
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print ' %s [label="%s' % (nodename, label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s"];' % ast[1]
        else:
            print '"]'
    else:
        print '"];'
        children = []
        for n, child in enumerate(ast[1:]):
            children.append(dotwrite(child))
        print ' %s -> {' % nodename,
        for name in children:
            print '%s' % name,
```

۱. ۷. هوش مصنوعی

شاید این خبر را شنیده باشید: « رایانه توانست گری کاسپاروف، مرد اول شطرنج جهان را شکست دهد ». شطرنج، یکی از پیچیده‌ترین بازی‌های فکری است و برنده شدن در آن، نتیجه‌ی مستقیم هوش فرد بازیکن می‌باشد. وقتی به دو عبارت بالا نگاه می‌کنیم، طبیعتاً نتیجه می‌گیریم که رایانه یک موجود هوشمند است. آیا این نکته به این معناست که انسان رقیبی سرسخت پیدا کرده است؟

گری کاسپاروف در حال مسابقه با دیپ جونیور در سال ۲۰۰۳

این مسابقه منجر به تساوی ۳-۳ گشت

بهتر است نگاهی به *الگوریتم* بازی شطرنج بیاندازیم. هنگام بازی، حریف با هر حرکت خود، ما را در موقعیتی جدید قرار می‌دهد. در این موقعیت جدید، ما امکان حرکات مختلفی را داریم که می‌توان تمام احتمالات ممکن را بر روی کاغذ نوشت. تعداد این حالات ممکن است زیاد باشد، ولی وقتی کمی دقت می‌کنیم متوجه می‌شویم که حریف هم به ازای هر یک از این حالات، امکان حرکات مختلفی را دارد. پس تعداد کل حرکات احتمالی در این دو مرحله بسیار زیاد می‌شود. حال شما این بحث را در مورد تمام مراحل یک بازی کامل شطرنج گسترش دهید. خواهید دید احتمالات ممکن را باید با ارقام نجومی بیان

کرد! در واقع راه صحیح شطرنج بازی کردن این است که ما در هر مرحله به ازای هر حرکت خود، بتوانیم حرکات حریف را در مراحل بعدی بسنجیم و بهترین حالت را انتخاب کنیم. وقتی بهترین حالت حاصل خواهد شد که ما بتوانیم تمام احتمالات ممکن را تا انتهای یک بازی پیش‌بینی کنیم. واضح است که این کار عملاً غیر ممکن است.

روش بالا روشی است که هر شطرنج بازی در ذهن خود اجرا می‌کند. بازیکنان مبتدی می‌توانند یک یا دو مرحله بعد را پیش بینی کنند و هرچه بازیکن حرفه‌ای‌تر باشد، می‌تواند مراحل بیشتری را پیش‌بینی کند. رایانه هم با همین روش شطرنج بازی می‌کند. البته به علت سرعت بالای محاسبات، رایانه می‌تواند در زمان کوتاهی مراحل بعدی بازی را پیش‌بینی کند.

می‌بینیم که رایانه برای انجام هر کاری، اگرچه به ظاهر هوشمندانه باشد، از یک الگوریتم مشخص پیروی می‌کند. به این الگوریتم‌ها که باعث می‌شوند رایانه رفتاری (به ظاهر) هوشمندانه داشته باشد، *الگوریتم‌های هوش مصنوعی* گفته می‌شود. هوش مصنوعی یکی از جالب‌ترین و پر کاربردترین حوزه‌های علم نرم‌افزار است.

الگوریتم‌های هوش مصنوعی، قابلیت حل مسائل پیچیده در زمان کوتاه را برای ما فراهم می‌کنند. هم‌چنین ما می‌توانیم با استفاده از این الگوریتم‌ها، قابلیت یادگیری را به نرم‌افزار و سخت‌افزارمان اضافه کنیم، برای مثال رباتی بسازیم که دستورات صاحبش را یاد بگیرد !