پندار C++

آلن بی. دوونی

کیارش بختیار

حق تکثیر

یادداشت مترجم

شاید اغراق نباشد که بگویم انگیزه‌ام از ترجمه این کتاب بیشتر خودخواهانه بوده‌است. از آن‌جا که به زبان برنامه نویسی C++ علاقه‌مند هستم و قصد یادگیری آن‌را داشتم، در گشت و گذار اینترنتی هنگامی که به دنبال منابع بودم با کتاب پیش رو آشنا شدم. همانگونه که در بخش مجوز مشاهده کردید، انتشار و ترجمه آن آزاد است. در نتیجه برآن شدم تا روش یادگیری حین ترجمه را تجربه کنم. از طرفی تصمیم گرفتم همسو با نویسنده، در نگارش کتاب نرم‌افزار قدرتمند حروف‌چینی لاتِکْ (LaTeX)را با استفاده از بسته زی‌پرشین (XePersian) به‌کار گیرم که جذابیت خاص خود را دارد. در نهایت برای کنترل منبعِ متن ترجمه، از نرم‌افزار git و همسان‌سازی آن با github در آدرس ذیل استفاده نمودم.

https://github.com/AllenDowney/ThinkCPP

کیارش بختیار، زمستان ۱۴۰۲

فهرست مطالب

حق تکثیر ۱
یادداشت مترجم ۲

۱ راه برنامه **۱**
زبان برنامه نویسی جدید چیست؟ ۱
What is a program؟ ۳
What is debugging؟ ۳
Compile-time errors ۳
Run-time errors ۴
Logic and semantics errors ۴
Experimental debugging ۴
natural and Formal languages ۵
The first program ۶
Glossary ۸

۲ Quick reference for AP classes **۹**
apstring ۹
apvector ۱۰
apmatrix ۱۱

فصل ۱

راه برنامه

هدف این کتاب این است که به شما بیاموزد مثل یک دانشمند کامپیوتر فکر کنید. من طرز فکر دانشمندان کامپیوتر را دوست دارم زیرا آنها برخی از بهترین ویژگی‌های ریاضیات، مهندسی و علوم طبیعی را باهم ترکیب می‌کنند. مانند ریاضیدانان دانشمندان کامپیوتر از زبان‌های رسمی برای نشان دادن ایده‌ها (به ویژه محاسبات) استفاده می‌کنند. آنها مثل مهندسان چیزهایی را طراحی می‌کنند، اجزای سازنده را در سامانه‌ها نصب می‌کنند و انتخاب یا سبک سنگین بین گزینه‌ها را ارزیابی می‌کنند. آنها همچون دانشمندان، رفتار سامانه‌های پیچیده را مشاهده می‌کنند سپس فرضیه‌ها را تشکیل می‌دهند و پیش بینی‌ها را آزمایش می‌کنند.
مهم‌ترین مهارت یک دانشمند کامپیوتر، **حل مسئله** است. منظور من توانایی در فرمول‌بندی مسائل، تفکر خلاقانه در مورد راه‌حل‌هاو بیان راه‌حل واضح و دقیق است. همانطور که مشخص است، فرآیند یادگیری برنامه‌نویسی فرصتی عالی برای تمرین مهارت‌های حل مسئله است. به همین دلیل این فصل «راه برنامه» نامیده می شود.

البته هدف دیگر این کتاب این است که شما را برای آزمون AP علوم کامپیوتر آماده کند. اگرچه ممکن است مستقیم ترین رویکرد را برای آن هدف در پیش نگیریم. به عنوان مثال، تمرین های زیادی در این کتاب وجود ندارد که شبیه به سوالات AP باشد. از سوی دیگر، اگر مفاهیم این کتاب را به همراه جزئیات برنامه نویسی در C++ درک کنید، تمام ابزارهایی را که برای موفقیت در امتحان نیاز دارید در اختیار خواهید داشت.

زبان برنامه نویسی جدید چیست؟

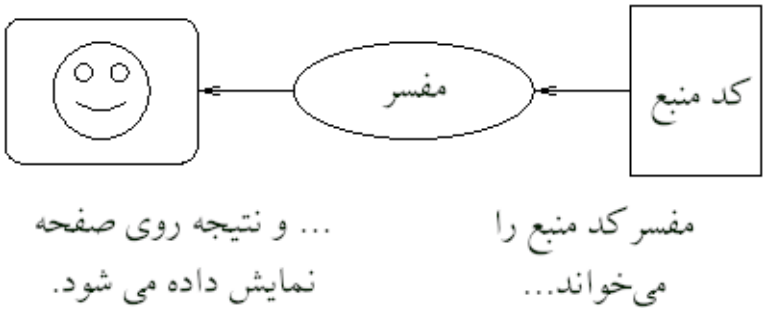
زبان برنامه نویسی که شما یاد خواهید گرفت C++ است، زیرا این زبانی است که آزمون AP از سال ۱۹۹۸ بر آن مبتنی است. قبل از آن، آزمون از Pascal استفاده می کرد. هر دو C++ و Pascal زبان های **سطح بالا** هستند. سایر زبان های سطح بالا که ممکن است نام آنها را شنیده باشید جاوا، سی و فرترن هستند.

همانطور که ممکن است از نام «زبان سطح بالا» استنباط کنید، زبان‌های **سطح پایین**، نیز وجود دارند. که گاهی به عنوان زبان ماشین یا زبان اسمبلی نامیده می شوند. به زبان ساده، کامپیوترها فقط می توانند برنامه‌هایی را که به زبان‌های سطح پایین نوشته شده‌اند اجرا کنند. بنابراین، برنامه‌های نوشته شده به یک زبان سطح بالا باید قبل از اجرا ترجمه شوند. این ترجمه مدتی طول می‌کشد که این یک نقطه ضعف کوچک برای زبان های سطح بالا است.

اما مزایای آن بسیار زیاد است. اولاً، برنامه نویسی در یک زبان سطح بالا *بسیار* ساده تر است. منظور من از «آسان تر» این است که نوشتن برنامه زمان کمتری می برد، خواندن آن کوتاه تر و راحت تر است و احتمال درستی آن بیشتر است. ثانیاً، زبان‌های سطح بالا **قابل حمل** هستند، به این معنی که می‌توانند بر روی انواع مختلف رایانه‌ها با تغییرات اندک یا بدون تغییر اجرا شوند. برنامه های سطح پایین فقط می توانند روی یک نوع کامپیوتر اجرا شوند و برای اجرا روی دیگری باید بازنویسی شوند.

با توجه به این مزایا، تقریباً همه برنامه ها به زبان های سطح بالا نوشته می شوند. زبان های سطح پایین فقط برای چند برنامه خاص استفاده می شوند.

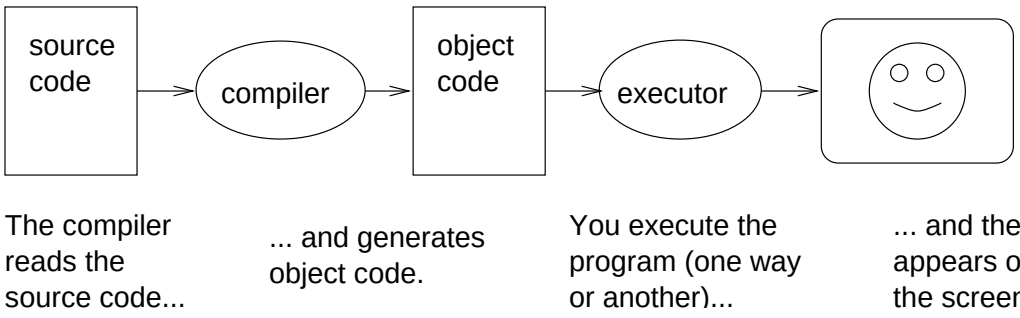
دو روش برای ترجمه برنامه وجود دارد. **تفسیر** یا **تدوین** مترجم برنامه ای است که یک برنامه سطح بالا را می‌خواند و آنچه را که می گوید انجام می دهد. در واقع، برنامه را خط به خط ترجمه می کند، متناوب خطوط را می خواند و دستورات را انجام می دهد.



executing before once، at all it translates and program high-level a reads that program a is compiler A the execute then and step، separate a as program the compile you Often commands. the of any translated the and ،**code source** the called is program high-level the case، this In later. code compiled .**executable** the or **code object** the called is program

program the write to editor text a use might You C++. in program a write you suppose example، an As file a in it save might you finished، is program the When processor). word simple a is editor text (a a is .cpp suffix the and up، make you name arbitrary an is “program” where ،`program.cpp` named code. source C++ contains file the that indicates that convention

and editor text the leave might you like، is environment programming your what on depending Then، named file new a create and it، translate code، source your read would compiler The compiler. the run executable. the contain to `program.exe` or code، object the contain to `program.o`



is executor the of role The executor. of kind some requires which program، the run to is step next The the executing start computer the make and memory) into disk from it (copy program the load to program.

programming most in that is news good the complicated، seem may process this Although you. for automated are steps these environments)، development called (sometimes environments On it. run and compile to command single a type and program a write to have only will you Usually if that so background، the in happening are that are steps the what know to useful is it hand، other the

is. it what out figure can you wrong goes something

program؟ a is What

The computation. a perform to how specifies that instructions of sequence a is program A roots the finding or equations of system a solving like mathematical، something be might computation a in text replacing and searching like computation، symbolic a be also can it but polynomial، a of program. a compiling enough) (strangely or document

languages، programming different in different look statements) or commands، (or instructions The language: every about just in appear that functions basic few a are there but

device. other some or file، a or keyboard، the from data Get **input:**

device. other or file a to data send or screen the on data Display **output:**

multiplication. and addition like operations mathematical basic Perform **math:**

statements. of sequence appropriate the execute and conditions certain for Check **testing:**

variation. some with usually repeatedly، action some Perform **repetition:**

how matter no used، ever you've program Every it. to is there all much pretty that's not، or it Believe describe to way one Thus، these. like less or more look that functions of up made is complicated، subtasks smaller and smaller into up task complex large، a breaking of process the is programming functions. simple these of one with performed be to enough simple are subtasks the eventually until

debugging؟ is What

For errors. to leads often it beings، human by done is it since and process، complex a is Programming and down them tracking of process the and **bugs** called are errors programming reasons، whimsical .**debugging** called is them correcting

distinguish to useful is it and program، a in occur can that errors of kinds different few a are There quickly. more down them track to order in them between

errors Compile-time

the otherwise، correct؛ syntactically is program the if program a translate only can compiler The your of structure the to refers **Syntax** program. your run to able be not will you and fails compilation structure. that about rules the and program

this period. a with end and letter capital a with begin must sentence a English، in example، For one this does So error. syntax a contains sentence

the read can we why is which problem، significant a not are errors syntax few a readers، most For messages. error spewing without cummings e e of poetry

the program، your in anywhere error syntax single a is there If forgiving. so not are Compilers program. your run to able be not will you and quit، and message error an print will compiler

error the and English، in are there than C++ in rules syntax more are there worse، matters make To your of weeks few first the During helpful. very not often are compiler the from get you messages gain you As errors. syntax down tracking time of lot a spend probably will you career، programming faster. them find and errors fewer make will you though، experience،

errors Run-time

you until appear not does error the because so-called error، run-time a is error of type second The program. the run

so rare، are errors run-time weeks، few next the for writing be will we programs of sorts simple the For one. encounter you before while little a be might it

semantics and errors Logic

it program، your in error logical a is there If error. **semantic** or **logical** the is error of type third The messages، error any generate not will computer the that sense the in successfully، run and compile will do. to it told you what do will it Specifically، else. something do will It thing. right the do not will it but

of meaning The write. to wanted you program the not is wrote you program the that is problem The to you requires it since tricky، be can errors logical Identifying wrong. is semantics) (its program the doing. is it what out figure to trying and program the of output the at looking by backwards work

debugging Experimental

debugging. is book this with working from acquire should you skills important most the of One and challenging، rich، intellectually most the of one is debugging frustrating، be can it Although programming. of parts interesting

infer to have you and clues with confronted are You work. detective like is debugging ways some In see. you results the to lead that events and processes the

you wrong، going is what idea an have you Once science. experimental an like also is Debugging of result the predict can you then correct، was hypothesis your If again. try and program your modify you wrong، was hypothesis your If program. working a to closer step a take you and modification، the the eliminated have you “When out، pointed Holmes Sherlock As one. new a with up come to have *The* Doyle’s Conan A. (from truth.” the be must improbable، however remains، whatever impossible، .(*Four of Sign*

the is programming is، That thing. same the are debugging and programming people، some For should you that is idea The want. you what does it until program a debugging gradually of process debugging modifications، small make and *something*، does that program working a with start always program. working a have always you that so go، you as them

out started it but code، of lines of thousands contains that system operating an is Linux example، For Larry to According chip. ۸۰۳۸۶ Intel the explore to used Torvalds Linus program simple a as AAAA printing between switch would that program a was projects earlier Linus’s of “One Greenfield، .(۱ Version Beta *Guide Users’ Linux The* (from Linux” to evolved later This BBBB. and

practices. programming other and debugging about suggestions more make will I chapters later In

languages natural and Formal

were They French. and Spanish، English، like speak، people that languages the are **languages Natural** naturally. evolved they them)؛ on order some impose to try people (although people by designed not

example، For applications. specific for people by designed are that languages are **languages Formal** denoting at good particularly is that language formal a is use mathematicians that notation the chemical the represent to language formal a use Chemists symbols. and numbers among relationships importantly: most And molecules. of structure

com- express to designed been have that languages formal are languages Programming putations.

$3 + 3 = 6$ example، For syntax. about rules strict have to tend languages formal before، mentioned I As syntactically a is H_2O Also، not. is $3 = +6\$$ but statement، mathematical correct syntactically a is not. is ${}_2Zz$ but name، chemical correct

of elements basic the are Tokens structure. and tokens to pertaining flavors، two in come rules Syntax that is `3=+6$` with problems the of One elements. chemical and numbers and words like language، the because legal not is ${}_2Zz$ Similarly، know). I as far as least (at mathematics in token legal a not is $\$.Zz$ abbreviation the with element no is there

tokens the way the is، that statement؛ a of structure the to pertains error syntax of type second The sign plus a have can't you because illegal، structurally is `3=+6$` statement The arranged. are the after subscripts have to have formulas molecular Similarly، sign. equals an after immediately before. not name، element

what out figure to have you language، formal a in statement a or English in sentence a read you When This unconsciously). this do you language natural a in (although is sentence the of structure the .**parsing** called is process

other "the that understand you fell،" shoe other "The sentence، the hear you when example، For what out figure can you sentence، a parsed have you Once verb. the is "fell" and subject the is shoe" it what and is، shoe a what know you that Assuming sentence. the of semantics the is، that means، it sentence. this of implication general the understand will you fall، to means

and syntax structure، common—tokens، in features many have languages natural and formal Although differences. many are semantics—there

clues contextual using by with deal people which ambiguity، of full are languages Natural **ambiguity:** unambiguous، completely or nearly be to designed are languages Formal information. other and context. of regardless meaning، one exactly has statement any that means which

em- languages natural misunderstandings، reduce and ambiguity for up make to order In **redundancy:** redundant less are languages Formal verbose. often are they result، a As redundancy. of lots ploy concise. more and

is there fell،" shoe other "The say، I If metaphor. and idiom of full are languages Natural **literalness:** say. they what exactly mean languages Formal falling. nothing and shoe no probably

to adjusting time hard a have often (everyone) language natural a speaking up grow who People the like is language natural and formal between difference the ways some In languages. formal so: more but prose، and poetry between difference

together poem whole the and meaning، their for as well as sounds their for used are Words **Poetry:** deliberate. often but common only not is Ambiguity response. emotional or effect an creates

meaning. more contributes structure the and important more is words of meaning literal The **Prose:** ambiguous. often still but poetry، than analysis to amenable more is Prose

understood be can and literal، and unambiguous is program computer a of meaning The **Programs:** structure. and tokens the of analysis by entirely

that remember First، languages). formal other (and programs reading for suggestions some are Here Also، them. read to longer takes it so languages، natural than dense more much are languages formal right. to left bottom، to top from read to idea good a not usually is it so important، very is structure the structure. the interpreting and tokens the identifying head، your in program the parse to learn Instead، punctuation، bad and errors spelling like things Little matter. details the that remember Finally، language. formal a in difference big a make can languages، natural in with away get can you which

program first The

it all because World." "Hello، called is language new a in write people program first the Traditionally this: like looks program this C++، In World." "Hello، words the print is does

```
<maertsoi> edulcni#
؛dts ecapseman gnisu

tuptuo elpmis emos etareneg :niam //

() niam tni
}
؛ldne >> ".dlrow، olleH" >> tuoc
؛۰ nruter
{
```

World." "Hello، the of simplicity the by language programming a of quality the judge people Some several contains program simple this so، Even well. reasonably does C++ standard، this By program. them، of some ignore will we now، For programmers. beginning to explain to hard are that features lines. two first the like

text English of bit a is comment A .**comment** a is it that indicates which ،// with begins line third The the When does. program the what explain to usually program، a of middle the in put can you that line. the of end the until there from everything ignores it ،// a sees compiler

special a is `main` .`main` word the notice but now، for `int` word the ignore can you line، fourth the In it runs، program the When begins. execution where program the in place the indicates that name last the to gets it until order، in continues، it and `main` in statement first the executing by starts quits. it then and statement،

one. only contains example the but ،`main` in be can that statements of number the to limit no is There screen. the on message a displays or outputs it that meaning statement، **output** basic a is It

symbol The screen. the to output send to you allow to system the by provided object special a is `cout` displayed. be to string the causes that and string، a and `cout` to apply you that **operator** an is <<

causes it ،`cout` to `endl` an send you When line. a of end the represents that symbol special a is `endl` text new the something، output you time next The display. the of line next the to move to cursor the line. next the on appears

.(;) semi-colon a with ends statement output the statements، all Like

uses C++ First، program. this of syntax the about notice should you things other few a are There in enclosed is statement output the case، this In together. things group to ({ and }) squiggly-braces is statement the that notice Also، .`main` of definition the *inside* is it that indicating squiggly-braces، definition. the inside are lines which visually show to helps which indented،

this run and compile and computer a of front in down sit to idea good a be would it point this At on now from but environment، programming your on depend that do to how of details The program. it. do to how know you that assume will I book this in

type you when errors any make you If syntax. for stickler real a is compiler C++ the mentioned، I As misspell you if example، For successfully. compile not will it that are chances program، the in following: the like message error an get might you ،`iostream`

```
directory or file such No oistream.h: hello.cpp:1:
```

to easy not is that format dense a in presented is it but line، this on information of lot a is There like: something say might compiler friendly more A interpret.

named file header a include to tried you hello.cpp، named file code source the of ١ line “On iostream. named something find did I but name، that with anything find didn’t I oistream.h. chance؟” any by meant، you what that Is

most in and smart، very really not is compiler The accomodating. so are compilers few Unfortunately، gain to time some take will It wrong. is what about hint a only be will get you message error the cases messages. compiler interpreting at facility

Starting language. a of rules syntax the learning for tool useful a be can compiler the Nevertheless، get you If happens. what see and ways various in it modify hello.cpp)، (like program working a with in again it see you if so it، caused what and says message the what remember to try message، error an means. it what know will you future the

Glossary

problem-solving: The formulating of a problem, finding a solution, and expressing the solution.

language: high-level A programming language like C++ that is designed to be easy for humans to read and write.

language: low-level A programming language that is designed to be easy for a computer to execute. Also called “machine language” or “assembly language.”

portability: A property of a program that can run on more than one kind of computer.

language: formal Any of the languages people have designed for specific purposes, like representing mathematical ideas or computer programs. All programming languages are formal languages.

language: natural Any of the languages people speak that have evolved naturally.

interpret: To execute a program in a high-level language by translating it one line at a time.

compile: To translate a program in a high-level language into a low-level language, all at once, in preparation for later execution.

code: source A program in a high-level language before being compiled.

code: object The output of the compiler after translating the program.

executable: Another name for object code that is ready to be executed.

algorithm: A general process for solving a category of problems.

bug: An error in a program.

syntax: The structure of a program.

semantics: The meaning of a program.

parse: To examine a program and analyze the syntactic structure.

error: syntax An error in a program that makes it impossible to parse (and therefore impossible to compile).

error: run-time An error in a program that makes it fail at run-time.

error: logical An error in a program that makes it do something other than what the programmer intended.

debugging: The process of finding and removing any of the three kinds of errors.

پیوست آ

classes AP for reference Quick

page، web Board College the from copied are definitions class These

-science/html/quick_ref.htmhttp://www.collegeboard.org/ap/computer

the from also text، following the repeat to time good a probably is This changes. formatting minor with page. web Board College

Sci- Computer Placement Advanced the in use for defined classes C++ the of "Inclusion by textbook this in material other the of endorsement constitute not does courses ence Development Science Computer AP the or service، Testing Educational Board، College the Science Computer AP the in use for defined classes C++ the of versions The Committee. classes the to Revisions .۱۹۹۹ July ۲۰ of as accurate were textbook this in included courses time." that since made been have may

apstring

```
string the in position a not indicate to used //  npos; int const extern

                                                functions member public //

                                              constructors/destructor //
      "" string empty construct //                          apstring();
literal string from construct //          s); * char apstring(const
            constructor copy //    str); & apstring apstring(const
                 destructor //                          ~apstring();

                                                          assignment //
str assign // str); & apstring (const operator= & apstring const
  s assign //       s); * char (const operator= & apstring const
 ch assign //              ch); (char operator= & apstring const
```

```
                                                                accessors //
                    chars of number //                      const; length() int
  str of occurrence first of index //    const; str) & apstring find(const int
   ch of occurrence first of index //                 const; ch) find(char int
          ,chars len of substring // const; len) int ,pos substr(int apstring
                  pos at starting //
    * char to conversion explicit //              const; c_str() * char const

                                                                 indexing //
                     indexing -checkedrange // const; k) ](int operator[ char
                     indexing -checkedrange //     k); ](int operator[ & char

                                                               modifiers //
           str append // str); & apstring (const operator+= & apstring const
          char append //              ch); (char operator+= & apstring const

     strings on operate functions -member)(non free following The //

                                                            functions I/O //
        ); str & apstring const ,os & ostream ( <<operator & ostream
              ); str & apstring ,is & istream ( >>operator & istream
                  ); str & apstring ,is & istream getline( & istream

                                                     operators comparison //
     ); rhs & apstring const ,lhs & apstring const ( operator== bool
     ); rhs & apstring const ,lhs & apstring const ( operator!= bool
     ); rhs & apstring const ,lhs & apstring const (  <operator bool
     ); rhs & apstring const ,lhs & apstring const ( <=operator bool
     ); rhs & apstring const ,lhs & apstring const (  >operator bool
     ); rhs & apstring const ,lhs & apstring const ( >=operator bool

                                                 + operator concatenation //
  ); rhs & apstring const ,lhs & apstring const ( operator+ apstring
              ); str & apstring const ,ch char ( operator+ apstring
              ); ch char ,str & apstring const ( operator+ apstring
```

apvector

```
                                                    >itemType <class template
                                                              apvector class

                                                  functions member public //

                                                 constructors/destructor //
```

```
 (size==0) constructor default //                                  apvector();
size is vector of size initial //                          size); apvector(int
 fillValue == entries all //  fillValue); & itemType const ,size apvector(int
             constructor copy //            vec); & apvector apvector(const
                   destructor //                                  ~apvector();

                                                              assignment //
                          vec); & apvector (const operator= & apvector const

                                                              accessors //
         vector of capacity //                            const; length() int

                                                               indexing //
                                                 checking range with indexing //
                                              index); ](int operator[ & itemType
                                 const; index) ](int operator[ & itemType const

                                                              modifiers //
           dynamically size change //              newSize); resize(int void
        values losing in result //can
```

apmatrix

```
                                                >itemType <class template
                                                           apmatrix class

                                               functions member public //

                                             constructors/destructor //
      0 x 0 is size default //                                  apmatrix();
        cols x rows is size //                cols); int ,rows apmatrix(int
                   fillValue); & itemType const ,cols int ,rows apmatrix(int
    fillValue == entries all //
            constructor copy //               mat); & apmatrix apmatrix(const
                  destructor //                                ); ~apmatrix(

                                                              assignment //
                          rhs); & apmatrix (const = operator & apmatrix const

                                                               accessors //
       rows of number //                                 const; numrows() int
     columns of number //                                const; numcols() int

                                                               indexing //
```

```
                                                    indexing -checkedrange //
                                  const; k) ](int operator[ & ><itemTypeapvector const
                                              k); ](int operator[ & ><itemTypeapvector

                                                                          modifiers //
newCols x newRows to matrix resizes // newCols); int ,newRows resize(int void
      values) losing in result (can //
```