

Lecture 2: Basic C++ I/O and Expression

Class page: <https://github.com/tsung-wei-huang/cs1420>

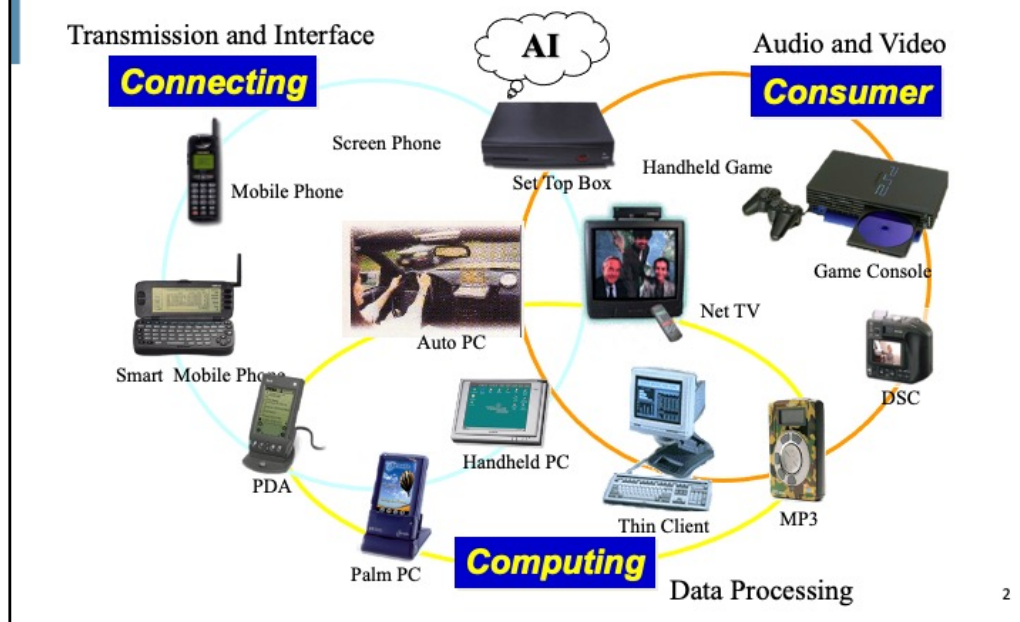
Dr. Tsung-Wei Huang
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT



Morning everyone

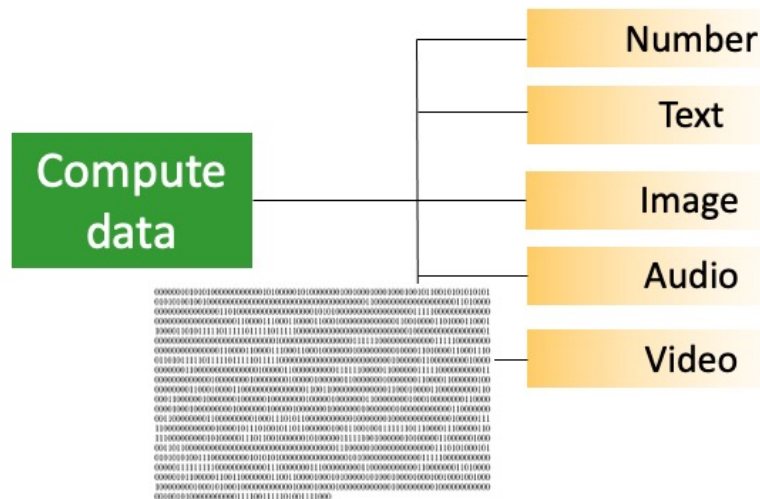
Today we are going to look at some basic C++ input and output and expression.

Our World is Driven by Computing



Last time, we talked about we are living in a highly connected world. The internet has become the knowledge hub for almost everyone in the world to get connected to each other and the newest trends. We use the internet to do almost all kinds of jobs. And a critical component to support this success is the computing. We want to compute data. we want to extract some useful information out of the data.

Computing Plays a Key Role



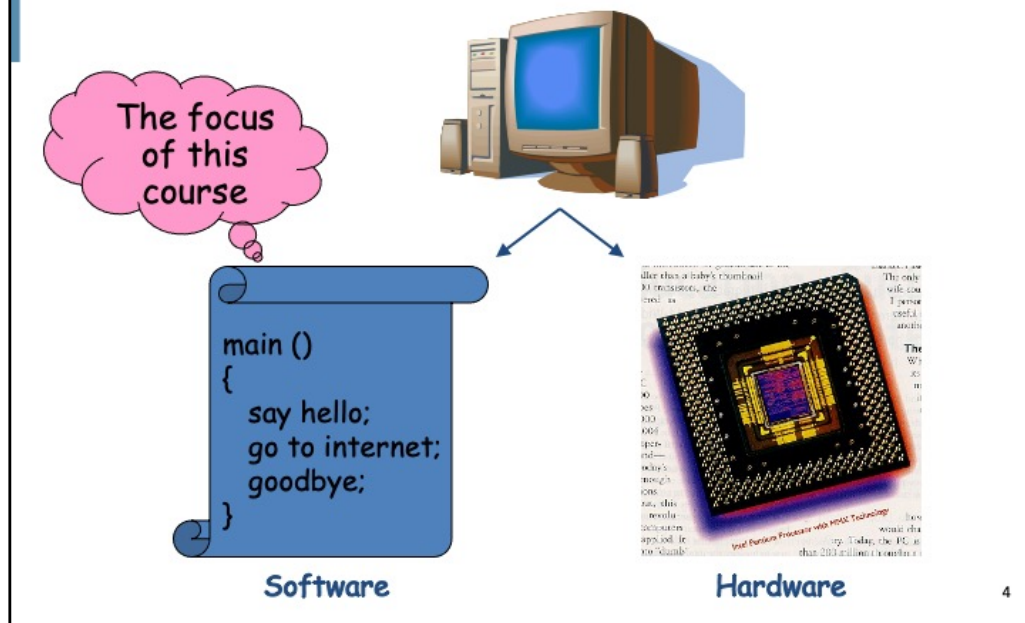
Computer see everything in binary (0 and 1)

3

Data can be numbers (like you phone number), or plain text, images (which are the most commonly used data), audio and video (like youtube). Whichever data you have, we need to compute the data to extract some useful information. We cannot do this by hand, because today's data is so huge and so enormous that we cannot rely on human to do all these things for us. We need to rely on computers because they can compute data much faster than human.

This is indeed very challenging. In the world of computers, everything is digitized. everything is just zero or one. That is what computers see everything in his world. Of course, we must avoid this non-human readable information because no one will be able to interpret zero and one.

Hardware vs Software

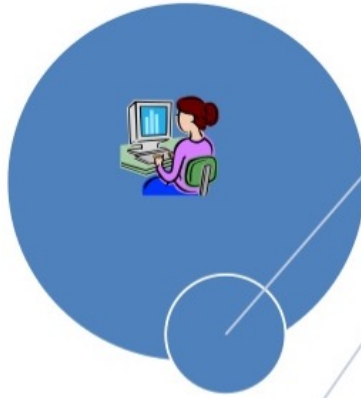


During the computation, there are two phases, one software and another one hardware. Software is the major layer for people to talk to computer. Hardware is the layer for computer to actually work on your program and run it on computing resources.

By the term you can probably guess, software is more flexible because when you get something wrong, you can just rewrite it, wipe it out and restart everything again.

However, it's a different story on the hardware. When you have a malfunction in hardware, it's million dollars. And the manufacturing process in today's hardware devices are so complicated that you won't afford to have too many errors. This is the fundamental difference between software and hardware.

What is Software?



Software is a general term for the various kinds of programs used to operate computers and related devices. (The term hardware describes the physical aspects of computers and related devices.)

What is software? ...

How do We Talk to Computers?

❑ Like we talk to human through “language”

- ❑ English, Mandarin, Hindi, ... => depending on countries
- ❑ C++, PHP, Python, JS, ... => depending on applications



6

In order to operate computers, we must first talk to them. But how do we talk to computers?

Like we talk to human, we use language. We talk to people from different countries using different languages, English, Mandarin, Hindi, and so on, it all depends on countries. For example, if the person comes from the US or the UK, we talk in English. If the person comes from China, we talk in Mandarin.

In computer programming, this "country aspect" translates to "applications". Depending on applications, we have different programming languages, such as C++, Php, python, and javascript. For instance, if you are doing data science, you need quick turnaround time and Python is the best choice. If you are doing computer engineering or electrical engineering, C or C++ is the right language. So, it really depends on the application you target. And the application is driven by the market.

Top 10 Programming Languages

| Rank | Language | Type | Score |
|------|------------|------|-------|
| 1 | Python | | 100.0 |
| 2 | Java | | 96.3 |
| 3 | C | | 94.4 |
| 4 | C++ | | 87.5 |
| 5 | R | | 81.5 |
| 6 | JavaScript | | 79.4 |
| 7 | C# | | 74.5 |
| 8 | Matlab | | 70.6 |
| 9 | Swift | | 69.1 |
| 10 | Go | | 68.0 |

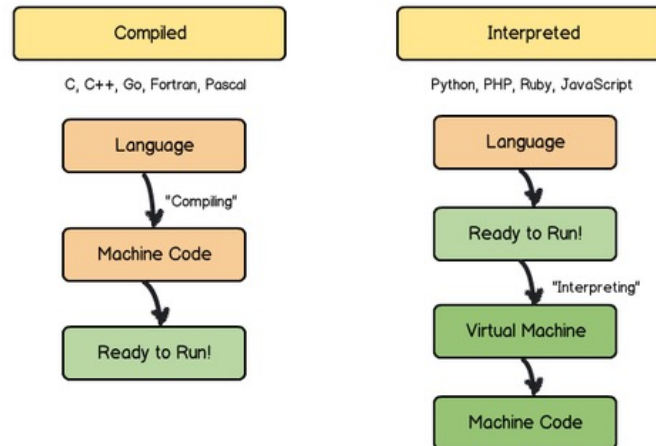
| | | | | | |
|----|-------------------|--|----|----------------------|--|
| 1 | Java | | 11 | MATLAB | |
| 2 | C | | 12 | R | |
| 3 | Python | | 13 | Perl | |
| 4 | C++ | | 14 | Assembly Language | |
| 5 | Visual Basic .NET | | 15 | Swift | |
| 6 | JavaScript | | 16 | Go | |
| 7 | C# | | 17 | Delphi/Object Pascal | |
| 8 | PHP | | 18 | Ruby | |
| 9 | SQL | | 19 | PL/SQL | |
| 10 | Objective-C | | 20 | Visual Basic | |

7

This slides shows the top 10 programming languages from multiple resources in 2019. Different tech reports have different results and some of the reports favor their preferred languages. But you can clearly see some of the languages appear in all reports, such as Java, Python, C, and C++, because they are just too important to be ignored. Languages like Java, Python, and C++ dominate the entire software industry.

Two Types of Programming Languages

❑ Compiled language vs Interpreted language



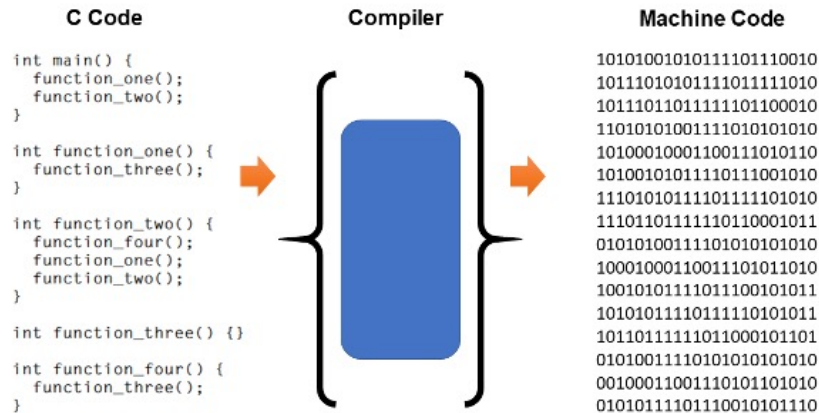
8

Roughly speaking, there are two types of programming languages: compiled language and interpreted language. The key difference is:

- the program written in compiled language cannot run directly (...)
- the program written in interpreted language can run directly (...)

Compilation (e.g., C/C++)

- ❑ Compile your “entire” code to an executable



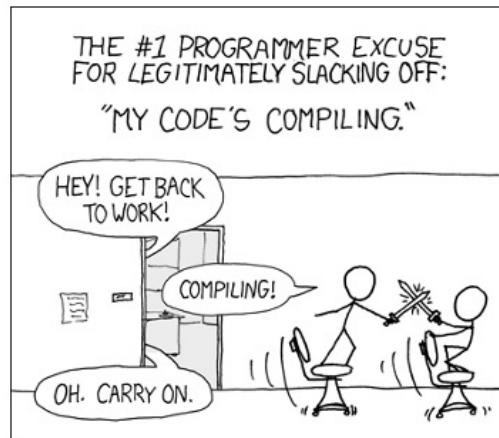
9

Let's look at an example of compilation using C programming language. On the left-hand side we have a source code written in C (you don't need to understand its syntax but just recognize it is a C program). In order to run the code, you need to have a compiler first, and invoke the compiler to compile the source code into machine-readable code. Machine-readable code is also called "assembly". Basically, they are just a bunch of zero and ones. These machine code are highly optimized based on your platform. You know, different platforms have different assembly architectures, such as Windows book, MACbook, Google chrome book, and the compiler can generate the machine code optimized for that platform.

So, a key take-away here is, compiled language can give you the best performance because the program you generated is highly optimized on a particular platform.

Compilation can be Very Long

- ❑ May take **HOURS** to compile a large codebase ...
 - ❑ Do you know how long it takes to compile Google code?



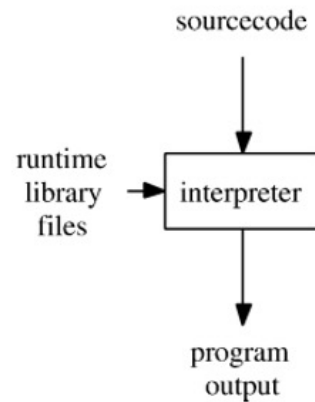
10

The major downside of compiled language is compilation can be very long. It may take hours to compile a large codebase ...

In a company I work before, if we want to find someone for coffee, we don't say "let's go for coffee" but "hey, get your code compiled". It's funny because it is really long, and everytime I change something I need to get it compiled first, before I can run it to test my results. This is the biggest problem of compiled language.

Interpretation

- ❑ Run your code line by line



11

Different from compilation, interpretation runs your code line by line. It doesn't need to see the entire code at once like compiled language but can run it line by line. This means, if there are syntax errors, like you write something wrong, the interpreter will not encounter that errors until it executes that line of code. So you have a much faster turnaround time to test your code during the development phase.

Interpretation can be Very Slow

Slowest things on earth:



12

You probably can guess, interpretation can be very slow - because it runs everything line-by-line without compilation that optimizes the control flow and code. Many people are joking with Python which is a very famous interpreted language, the speed is like turtle.

Pros and Cons

❑ Again, real cost depends on applications

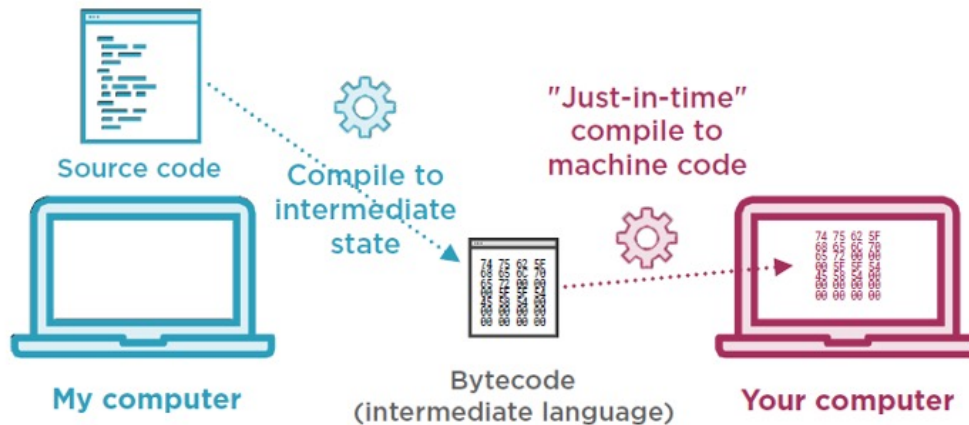
| Compiled | | Interpreted | |
|-------------------------------|---------------------------|-----------------|------------------------------|
| PROS | CONS | PROS | CONS |
| ready to run | not cross platform | cross-platform | interpreter required |
| often faster | inflexible | simpler to test | often slower |
| source code is private | extra step | easier to debug | source code is public |

13

This table summarizes the pros and cons between compiled language and interpreted language.

Just-in-time Compilation

- ❑ A mix of both (e.g., JavaScript)



14

In fact, there is a recent technology becoming more and more popular called Just-in-time compilation or JIT for short. Just in time compilation essentially combines the advantages of compiled language and interpreted language. What it does is, it starts with interpretation, and keeps running the code. While running the code, it does some analysis on the code and may invoke a native compiler to generate assembly to optimize some performance critical blocks. So you can see this type of technology is very useful for applications that have regular or iterative process. (explore...

Of course, Just in time compilation also inherits the downsides of both, so it is used in certain applications and is not replacing interpreter or compiler.

What Types of Language you will use?

- ☐ I am doing embedded software for autonomous cars
- ☐ I am working on performance-critical applications
- ☐ I am building a high-frequency trading platform
- ☐ I am prototyping my software products
- ☐ I am building circuit design software
- ☐ I am creating my own website

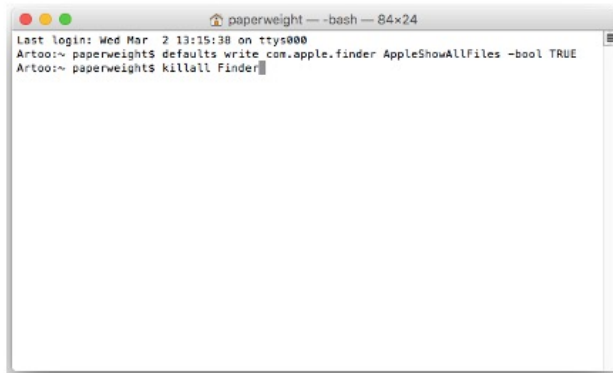
15

Let's stop here for 3 minutes and please take a look at the following scenarios and think about which kind of language you may need to use.

Programming Environment

❑ Terminal

- ❑ A computer terminal is an electronic or electromechanical hardware device that can be used for *entering data into*, and displaying or printing data from, a computer or a computing system.



16

Now we know the two types of programming language, what about the programming environment. In this course, we will use "terminal" to write your program. A computer terminal is an

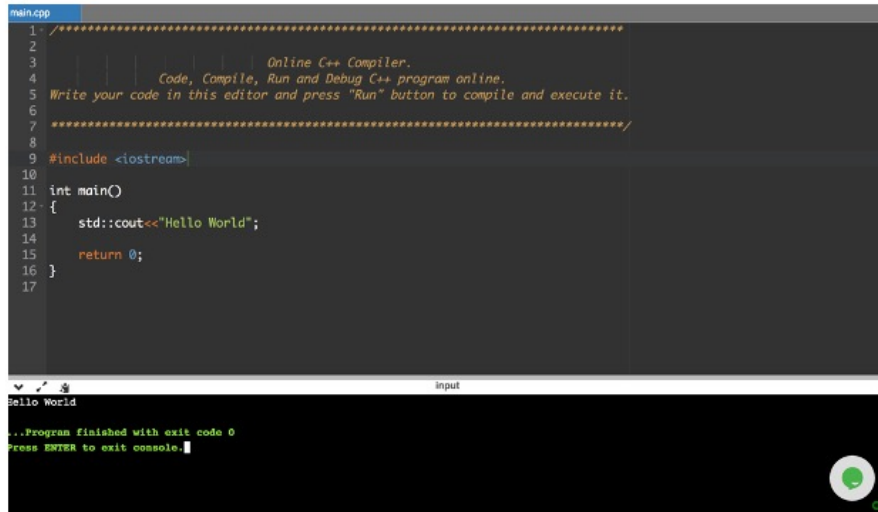
Essentially, a terminal is the "most basic" interface for you to operate computer without graphic interface. It is very simple and we will use terminal to write programs most of time.

Editor (I normally use VIM)

```
61 --- 3 lines: Is a socket 'connection oriented' ?-----[211] EXPORT_SYMBOL(netpoll_poll_dev);
62 static inline int connection_based(struct sock *sk) [212]
63 --- 3 lines: {-----[213] void netpoll_poll(struct netpoll *np)
64 [214] --- 3 lines: {-----
65 static int receiver_wake_function(wait_queue_t *wait, unsigned [217] EXPORT_SYMBOL(netpoll_poll);
66 void *key) [218]
67 --- 10 lines: {-----[219] static void refill_skbs(void)
68 [220] --- 3 lines: Wait for a packet,-----[221] 14 lines: {-----
69 static int wait_for_packet(struct sock *sk, int *err, long *tim [222]
70 [223] --- 45 lines: {-----[224]
71 [225] static void zap_completion_queue(void)
72 [226] --- 26 lines: {-----
73 [227]
74 [228] static struct sk_buff *find_skb(struct netpoll *np, int len, int
75 [229] --- 24 lines: {-----
76 [230] static int netpoll_owner_active(struct net_device *dev)
77 [231] --- 9 lines: {-----
78 [232]
79 [233] void netpoll_send_skb_on_dev(struct netpoll *np, struct sk_buff *
80 [234] struct net_device *dev)
81 [235] --- 54 lines: {-----
82 EXPORT_SYMBOL(netpoll_send_skb_on_dev);
83 [236]
84 void netpoll_send_udp(struct netpoll *np, const char *msg, int le [237]
85 [238] --- 59 lines: {-----
86 EXPORT_SYMBOL(netpoll_send_udp);
87 [239]
88 [240] static void arp_reply(struct sk_buff *skb)
89 [241] --- 116 lines: {-----
90 [242]
91 int netpoll_rx(struct sk_buff *skb)
92 [243] --- 90 lines: {-----
93 [244]
94 void netpoll_print_options(struct netpoll *np)
95 [245] --- 14 lines: {-----
96 EXPORT_SYMBOL(netpoll_print_options);
97 [246]
98 [247]
99 [248]
100 [249]
101 [250]
102 [251]
103 [252]
104 [253]
105 [254]
106 [255]
107 [256]
108 [257]
109 [258]
110 [259]
111 [260]
112 [261]
113 [262]
114 [263]
115 [264]
116 [265]
117 [266]
118 [267]
119 [268]
120 [269]
121 [270]
122 [271]
123 [272]
124 [273]
125 [274]
126 [275]
127 [276]
128 [277]
129 [278]
130 [279]
131 [280]
132 [281]
133 [282]
134 [283]
135 [284]
136 [285]
137 [286]
138 [287]
139 [288]
140 [289]
141 [290]
142 [291]
143 [292]
144 [293]
145 [294]
146 [295]
147 [296]
148 [297]
149 [298]
150 [299]
151 [300]
152 [301]
153 [302]
154 [303]
155 [304]
156 [305]
157 [306]
158 [307]
159 [308]
160 [309]
161 [310]
162 [311]
163 [312]
164 [313]
165 [314]
166 [315]
167 [316]
168 [317]
169 [318]
170 [319]
171 [320]
172 [321]
173 [322]
174 [323]
175 [324]
176 [325]
177 [326]
178 [327]
179 [328]
180 [329]
181 [330]
182 [331]
183 [332]
184 [333]
185 [334]
186 [335]
187 [336]
188 [337]
189 [338]
190 [339]
191 [340]
192 [341]
193 [342]
194 [343]
195 [344]
196 [345]
197 [346]
198 [347]
199 [348]
200 [349]
201 [350]
202 [351]
203 [352]
204 [353]
205 [354]
206 [355]
207 [356]
208 [357]
209 [358]
210 [359]
211 [360]
212 [361]
213 [362]
214 [363]
215 [364]
216 [365]
217 [366]
218 [367]
219 [368]
220 [369]
221 [370]
222 [371]
223 [372]
224 [373]
225 [374]
226 [375]
227 [376]
228 [377]
229 [378]
230 [379]
231 [380]
232 [381]
233 [382]
234 [383]
235 [384]
236 [385]
237 [386]
238 [387]
239 [388]
240 [389]
241 [390]
242 [391]
243 [392]
244 [393]
245 [394]
246 [395]
247 [396]
248 [397]
249 [398]
250 [399]
251 [400]
252 [401]
253 [402]
254 [403]
255 [404]
256 [405]
257 [406]
258 [407]
259 [408]
260 [409]
261 [410]
262 [411]
263 [412]
264 [413]
265 [414]
266 [415]
267 [416]
268 [417]
269 [418]
270 [419]
271 [420]
272 [421]
273 [422]
274 [423]
275 [424]
276 [425]
277 [426]
278 [427]
279 [428]
280 [429]
281 [430]
282 [431]
283 [432]
284 [433]
285 [434]
286 [435]
287 [436]
288 [437]
289 [438]
290 [439]
291 [440]
292 [441]
293 [442]
294 [443]
295 [444]
296 [445]
297 [446]
298 [447]
299 [448]
300 [449]
301 [450]
302 [451]
303 [452]
304 [453]
305 [454]
306 [455]
307 [456]
308 [457]
309 [458]
310 [459]
311 [460]
312 [461]
313 [462]
314 [463]
315 [464]
316 [465]
317 [466]
318 [467]
319 [468]
320 [469]
321 [470]
322 [471]
323 [472]
324 [473]
325 [474]
326 [475]
327 [476]
328 [477]
329 [478]
330 [479]
331 [480]
332 [481]
333 [482]
334 [483]
335 [484]
336 [485]
337 [486]
338 [487]
339 [488]
340 [489]
341 [490]
342 [491]
343 [492]
344 [493]
345 [494]
346 [495]
347 [496]
348 [497]
349 [498]
350 [499]
351 [500]
352 [501]
353 [502]
354 [503]
355 [504]
356 [505]
357 [506]
358 [507]
359 [508]
360 [509]
361 [510]
362 [511]
363 [512]
364 [513]
365 [514]
366 [515]
367 [516]
368 [517]
369 [518]
370 [519]
371 [520]
372 [521]
373 [522]
374 [523]
375 [524]
376 [525]
377 [526]
378 [527]
379 [528]
380 [529]
381 [530]
382 [531]
383 [532]
384 [533]
385 [534]
386 [535]
387 [536]
388 [537]
389 [538]
390 [539]
391 [540]
392 [541]
393 [542]
394 [543]
395 [544]
396 [545]
397 [546]
398 [547]
399 [548]
400 [549]
401 [550]
402 [551]
403 [552]
404 [553]
405 [554]
406 [555]
407 [556]
408 [557]
409 [558]
410 [559]
411 [560]
412 [561]
413 [562]
414 [563]
415 [564]
416 [565]
417 [566]
418 [567]
419 [568]
420 [569]
421 [570]
422 [571]
423 [572]
424 [573]
425 [574]
426 [575]
427 [576]
428 [577]
429 [578]
430 [579]
431 [580]
432 [581]
433 [582]
434 [583]
435 [584]
436 [585]
437 [586]
438 [587]
439 [588]
440 [589]
441 [590]
442 [591]
443 [592]
444 [593]
445 [594]
446 [595]
447 [596]
448 [597]
449 [598]
450 [599]
451 [600]
452 [601]
453 [602]
454 [603]
455 [604]
456 [605]
457 [606]
458 [607]
459 [608]
460 [609]
461 [610]
462 [611]
463 [612]
464 [613]
465 [614]
466 [615]
467 [616]
468 [617]
469 [618]
470 [619]
471 [620]
472 [621]
473 [622]
474 [623]
475 [624]
476 [625]
477 [626]
478 [627]
479 [628]
480 [629]
481 [630]
482 [631]
483 [632]
484 [633]
485 [634]
486 [635]
487 [636]
488 [637]
489 [638]
490 [639]
491 [640]
492 [641]
493 [642]
494 [643]
495 [644]
496 [645]
497 [646]
498 [647]
499 [648]
500 [649]
501 [650]
502 [651]
503 [652]
504 [653]
505 [654]
506 [655]
507 [656]
508 [657]
509 [658]
510 [659]
511 [660]
512 [661]
513 [662]
514 [663]
515 [664]
516 [665]
517 [666]
518 [667]
519 [668]
520 [669]
521 [670]
522 [671]
523 [672]
524 [673]
525 [674]
526 [675]
527 [676]
528 [677]
529 [678]
530 [679]
531 [680]
532 [681]
533 [682]
534 [683]
535 [684]
536 [685]
537 [686]
538 [687]
539 [688]
540 [689]
541 [690]
542 [691]
543 [692]
544 [693]
545 [694]
546 [695]
547 [696]
548 [697]
549 [698]
550 [699]
551 [700]
552 [701]
553 [702]
554 [703]
555 [704]
556 [705]
557 [706]
558 [707]
559 [708]
560 [709]
561 [710]
562 [711]
563 [712]
564 [713]
565 [714]
566 [715]
567 [716]
568 [717]
569 [718]
570 [719]
571 [720]
572 [721]
573 [722]
574 [723]
575 [724]
576 [725]
577 [726]
578 [727]
579 [728]
580 [729]
581 [730]
582 [731]
583 [732]
584 [733]
585 [734]
586 [735]
587 [736]
588 [737]
589 [738]
590 [739]
591 [740]
592 [741]
593 [742]
594 [743]
595 [744]
596 [745]
597 [746]
598 [747]
599 [748]
600 [749]
601 [750]
602 [751]
603 [752]
604 [753]
605 [754]
606 [755]
607 [756]
608 [757]
609 [758]
610 [759]
611 [760]
612 [761]
613 [762]
614 [763]
615 [764]
616 [765]
617 [766]
618 [767]
619 [768]
620 [769]
621 [770]
622 [771]
623 [772]
624 [773]
625 [774]
626 [775]
627 [776]
628 [777]
629 [778]
630 [779]
631 [780]
632 [781]
633 [782]
634 [783]
635 [784]
636 [785]
637 [786]
638 [787]
639 [788]
640 [789]
641 [790]
642 [791]
643 [792]
644 [793]
645 [794]
646 [795]
647 [796]
648 [797]
649 [798]
650 [799]
651 [800]
652 [801]
653 [802]
654 [803]
655 [804]
656 [805]
657 [806]
658 [807]
659 [808]
660 [809]
661 [810]
662 [811]
663 [812]
664 [813]
665 [814]
666 [815]
667 [816]
668 [817]
669 [818]
670 [819]
671 [820]
672 [821]
673 [822]
674 [823]
675 [824]
676 [825]
677 [826]
678 [827]
679 [828]
680 [829]
681 [830]
682 [831]
683 [832]
684 [833]
685 [834]
686 [835]
687 [836]
688 [837]
689 [838]
690 [839]
691 [840]
692 [841]
693 [842]
694 [843]
695 [844]
696 [845]
697 [846]
698 [847]
699 [848]
700 [849]
701 [850]
702 [851]
703 [852]
704 [853]
705 [854]
706 [855]
707 [856]
708 [857]
709 [858]
710 [859]
711 [860]
712 [861]
713 [862]
714 [863]
715 [864]
716 [865]
717 [866]
718 [867]
719 [868]
720 [869]
721 [870]
722 [871]
723 [872]
724 [873]
725 [874]
726 [875]
727 [876]
728 [877]
729 [878]
730 [879]
731 [880]
732 [881]
733 [882]
734 [883]
735 [884]
736 [885]
737 [886]
738 [887]
739 [888]
740 [889]
741 [890]
742 [891]
743 [892]
744 [893]
745 [894]
746 [895]
747 [896]
748 [897]
749 [898]
750 [899]
751 [900]
752 [901]
753 [902]
754 [903]
755 [904]
756 [905]
757 [906]
758 [907]
759 [908]
760 [909]
761 [910]
762 [911]
763 [912]
764 [913]
765 [914]
766 [915]
767 [916]
768 [917]
769 [918]
770 [919]
771 [920]
772 [921]
773 [922]
774 [923]
775 [924]
776 [925]
777 [926]
778 [927]
779 [928]
780 [929]
781 [930]
782 [931]
783 [932]
784 [933]
785 [934]
786 [935]
787 [936]
788 [937]
789 [938]
790 [939]
791 [940]
792 [941]
793 [942]
794 [943]
795 [944]
796 [945]
797 [946]
798 [947]
799 [948]
800 [949]
801 [950]
802 [951]
803 [952]
804 [953]
805 [954]
806 [955]
807 [956]
808 [957]
809 [958]
810 [959]
811 [960]
812 [961]
813 [962]
814 [963]
815 [964]
816 [965]
817 [966]
818 [967]
819 [968]
820 [969]
821 [970]
822 [971]
823 [972]
824 [973]
825 [974]
826 [975]
827 [976]
828 [977]
829 [978]
830 [979]
831 [980]
832 [981]
833 [982]
834 [983]
835 [984]
836 [985]
837 [986]
838 [987]
839 [988]
840 [989]
841 [990]
842 [991]
843 [992]
844 [993]
845 [994]
846 [995]
847 [996]
848 [997]
849 [998]
850 [999]
851 [1000]
852 [1001]
853 [1002]
854 [1003]
855 [1004]
856 [1005]
857 [1006]
858 [1007]
859 [1008]
860 [1009]
861 [1010]
862 [1011]
863 [1012]
864 [1013]
865 [1014]
866 [1015]
867 [1016]
868 [1017]
869 [1018]
870 [1019]
871 [1020]
872 [1021]
873 [1022]
874 [1023]
875 [1024]
876 [1025]
877 [1026]
878 [1027]
879 [1028]
880 [1029]
881 [1030]
882 [1031]
883 [1032]
884 [1033]
885 [1034]
886 [1035]
887 [1036]
888 [1037]
889 [1038]
890 [1039]
891 [1040]
892 [1041]
893 [1042]
894 [1043]
895 [1044]
896 [1045]
897 [1046]
898 [1047]
899 [1048]
900 [1049]
901 [1050]
902 [1051]
903 [1052]
904 [1053]
905 [1054]
906 [1055]
907 [1056]
908 [1057]
909 [1058]
910 [1059]
911 [1060]
912 [1061]
913 [1062]
914 [1063]
915 [1064]
916 [1065]
917 [1066]
918 [1067]
919 [1068]
920 [1069]
921 [1070]
922 [1071]
923 [1072]
924 [1073]
925 [1074]
926 [1075]
927 [1076]
928 [1077]
929 [1078]
930 [1079]
931 [1080]
932 [1081]
933 [1082]
934 [1083]
935 [1084]
936 [1085]
937 [1086]
938 [1087]
939 [1088]
940 [1089]
941 [1090]
942 [1091]
943 [1092]
944 [1093]
945 [1094]
946 [1095]
947 [1096]
948 [1097]
949 [1098]
950 [1099]
951 [1100]
952 [1101]
953 [1102]
954 [1103]
955 [1104]
956 [1105]
957 [1106]
958 [1107]
959 [1108]
960 [1109]
961 [1110]
962 [1111]
963 [1112]
964 [1113]
965 [1114]
966 [1115]
967 [1116]
968 [1117]
969 [1118]
970 [1119]
971 [1120]
972 [1121]
973 [1122]
974 [1123]
975 [1124]
976 [1125]
977 [1126]
978 [1127]
979 [1128]
980 [1129]
981 [1130]
982 [1131]
983 [1132]
984 [1133]
985 [1134]
986 [1135]
987 [1136]
988 [1137]
989 [1138]
990 [1139]
991 [1140]
992 [1141]
993 [1142]
994 [1143]
995 [1144]
996 [1145]
997 [1146]
998 [1147]
999 [1148]
1000 [1149]
1001 [1150]
1002 [1151]
1003 [1152]
1004 [1153]
1005 [1154]
1006 [1155]
1007 [1156]
1008 [1157]
1009 [1158]
1010 [1159]
1011 [1160]
1012 [1161]
1013 [1162]
1014 [1163]
1015 [1164]
1016 [1165]
1017 [1166]
1018 [1167]
1019 [1168]
1020 [1169]
1021 [1170]
1022 [1171]
1023 [1172]
1024 [1173]
1025 [1174]
1026 [1175]
1027 [1176]
1028 [1177]
1029 [1178]
1030 [1179]
1031 [1180]
1032 [1181]
1033 [1182]
1034 [1183]
1035 [1184]
1036 [1185]
1037 [1186]
1038 [1187]
1039 [1188]
1040 [1189]
1041 [1190]
1042 [1191]
1043 [1192]
1044 [1193]
1045 [1194]
1046 [1195]
1047 [1196]
1048 [1197]
1049 [1198]
1050 [1199]
1051 [1200]
1052 [1201]
1053 [1202]
1054 [1203]
1055 [1204]
1056 [1205]
1057 [1206]
1058 [1207]
1059 [1208]
1060 [1209]
1061 [1210]
1062 [1211]
1063 [1212]
1064 [1213]
1065 [1214]
1066 [1215]
1067 [1216]
1068 [1217]
1069 [1218]
1070 [1219]
1071 [1220]
1072 [1221]
1073 [1222]
1074 [1223]
1075 [1224]
1076 [1225]
1077 [1226]
1078 [1227]
1079 [1228]
1080 [1229]
1081 [1230]
1082 [1231]
1083 [1232]
1084 [1233]
1085 [1234]
1086 [1235]
1087 [1236]
1088 [1237]
1089 [1238]
1090 [1239]
1091 [1240]
1092 [1241]
1093 [1242]
1094 [1243]
1095 [1244]
1096 [1245]
1097 [1246]
1098 [1247]
1099 [1248]
1100 [1249]
1101 [1250]
1102 [1251]
1103 [1252]
1104 [1253]
1105 [1254]
1106 [1255]
1107 [1256]
1108 [1257]
1109 [1258]
1110 [1259]
1111 [1260]
1112 [1261]
1113 [1262]
1114 [1263]
1115 [1264]
1116 [1265]
1117 [1266]
1118 [1267]
1119 [1268]
1120 [1269]
1121 [1270]
1122 [1271]
1123 [1272]
1124 [1273]
1125 [1274]
1126 [1275]
1127 [1276]
1128 [1277]
1129 [1278]
1130 [1279]
1131 [1280]
1132 [1281]
1133 [1282]
1134 [1283]
1135 [1284]
1136 [1285]
1137 [1286]
1138 [1287]
1139 [1288]
1140 [1289]
1141 [1290]
1142 [1291]
1143 [1292]
1144 [1293]
1145 [1294]
1146 [1295]
1147 [1296]
1148 [1297]
1149 [1298]
1150 [1299]
1151 [1300]
1152 [1301]
1153 [1302]
1154 [1303]
1155 [1304]
1156 [1305]
1157 [1306]
1158 [1307]
1159 [1308]
1160 [1309]
1161 [1310]
1162 [1311]
1163 [1312]
1164 [1313]
1165 [1314]
1166 [1315]
1167 [1316]
1168 [1317]
1169 [1318]
1170 [1319]
1171 [1320]
1172 [1321]
1173 [1322]
1174 [1323]
1175 [1324]
1176 [1325]
1177 [1326]
1178 [1327]
1179 [1328]
1180 [1329]
1181 [1330]
1182 [1331]
1183 [1332]
1184 [1333]
1185 [1334]
1186 [1335]
1187 [1336]
1188 [1337]
1189 [1338]
1190 [1339]
1191 [1340]
1192 [1341]
1193 [1342]
1194 [1343]
1195 [1344]
1196 [1345]
1197 [1346]
1198 [1347]
1199 [1348]
1200 [1349]
1201 [1350]
1202 [1351]
1203 [1352]
1204 [1353]
1205 [1354]
1206 [1355]
1207 [1356]
1208 [1357]
1209 [1358]
1210 [1359]
1211 [1360]
1212 [1361]
1213 [1362]
1214 [1363]
1215 [1364]
1216 [1365]
1217 [1366]
1218 [1367]
1219 [1368]
1220 [1369]
1221 [1370]
1222 [1371]
1223 [1372]
1224 [1373]
1225 [1374]
1226 [1375]
1227 [1376]
1228 [1377]
1229 [1378]
1230 [1379]
1231 [1380]
1232 [1381]
1233 [1382]
1234 [1383]
1235 [1384]
1236 [1385]
1237 [1386]
1238 [1387]
1239 [1388]
1240 [1389]
1241 [1390]
1242 [1391]
1243 [1392]
1244 [1393]
1245 [1394]
1246 [1395]
1247 [1396]
1248 [1397]
1249 [1398]
1250 [1399]
1251 [1400]
1252 [1401]
1253 [1402]
1254 [1403]
1255 [1404]
1256 [1405]
1257 [1406]
1258 [1407]
1259 [1408]
1260 [1409]
1261 [1410]
1262 [1411]
1263 [1412]
1264 [1413]
1265 [1414]
1266 [1415]
1267 [1416]
1268 [1417]
1269 [1418]
1270 [1419]
1271 [1420]
1272 [1421]
1273 [1422]
1274 [1423]
1275 [1424]
1276 [1425]
1277 [1426]
1278 [1427]
1279 [1428]
1280 [1429]
1281 [1430]
1282 [1431]
1283 [1432]
1284 [1433]
1285 [1434]
1286 [1435]
1287 [1436]
1288 [1437]
1289 [1438]
1290 [1439]
1291 [1440]
1292 [1441]
1293 [1442]
1294 [1443]
1295 [1444]
1296 [1445]
1297 [1446]
1298 [1447]
1299 [1448]
1300 [1449]
1301 [1450]
1302 [1451]
1303 [1452]
1304 [1453]
1305 [1454]
1306 [1455]
1307 [1456]
1308 [1457]
1309 [1458]
1310 [1459]
1311 [1460]
131
```

There are also online C++ Editors

☐ https://www.onlinegdb.com/online_c++_compiler



The screenshot shows the Online GDB C++ compiler interface. The top section is a code editor with a dark background and light-colored text. It contains a C++ program that prints "Hello World". The code is as follows:

```
1. //=====
2.
3. Online C++ Compiler.
4. Code, Compile, Run and Debug C++ program online.
5. Write your code in this editor and press "Run" button to compile and execute it.
6.
7. =====
8.
9. #include <iostream>
10.
11. int main()
12. {
13.     std::cout<<"Hello World";
14.
15.     return 0;
16. }
17.
```

Below the code editor is a console window. It shows the output of the program, which is "Hello World". Below the output, it says "...Program finished with exit code 0" and "Press ENTER to exit console." There is a green circular button with a white play icon in the bottom right corner of the console window.

18

There are also online editor that you can use to write C++ code. This is useful when you don't have access to a terminal or a machine but you want to write C++ code and test it in a quick cycle.

Output - I

```
1
2 // Text-printing program.
3 #include <iostream> // allows program to output data to the screen
4
5 // function main begins program execution
6 int main()
7 {
8     std::cout << "Welcome to C++!\n"; // display message
9
10    return 0; // indicate that program ended successfully
11 } // end function main
```

Welcome to C++!

19

Any question so far?

Let's look at our first C++ program that prints out a simple hello message. This is a C++ code. The first thing you need to do is to include that allows your program to use functions that are provided by the libraries, instead of writing your own. Including a library is like you go to HomeDepot to buy a driller so you can use the driller to do construction work instead of building a driller on your own which is very difficult.

All the program starts from a main function. This is where the operating system recognizes the entry point of your program. When you run your program, the first entry point you enter is the main block.

Inside the main function, we use `std::cout` to output an message. `std::cout` is a global object defined by the library for you to output message. The message under the quotation indicates it is a string. A string is a human readable text.

Another important message here is '`std::`' represents the namespace. A namespace is where you can define your own things without naming conflict with others. Think about this ...

Finally, we return an integer from main function.

Output - II

- ❑ A single statement can print multiple lines by using newline characters.
- ❑ Each time the `\n` (newline) is encountered, the screen cursor is positioned to the beginning of the next line.

```
1
2 // Printing multiple lines of text with a single statement.
3 #include <iostream> // allows program to output data to the screen
4
5 // function main begins program execution
6 int main()
7 {
8     std::cout << "Welcome\n\tto\n\nC++!\n";
9 } // end function main
```

```
Welcome
to
C++!
```

20

For this printing statement using `std::cout`, a single state ...\

Header files #include <xxx>

Header files contain definitions of **Functions and Variables**, which is imported or used into any C++ program by using the pre-processor #include statement.

Header file have an extension ".h" which contains C++ function declaration and macro definition.

Why need of header files

When we want to use any function in our C++ program then first we need to import their definition from C++ library, for importing their declaration and definition we need to include header file in program by using #include.

Input - I

- ❑ The input stream object `std::cin` and the stream extraction operator, `>>`, can be used obtain data from the user at the keyboard.

```
1
2 // Addition program that displays the sum of two integers.
3 #include <iostream> // allows program to perform input and output
4
5 // function main begins program execution
6 int main()
7 {
8     // variable declarations
9     int number1; // first integer to add
10    int number2; // second integer to add
11    int sum; // sum of number1 and number2
12
13    std::cout << "Enter first integer: "; // prompt user for data
14    std::cin >> number1; // read first integer from user into number1
15
16    std::cout << "Enter second integer: "; // prompt user for data
17    std::cin >> number2; // read second integer from user into number2
18
19    sum = number1 + number2; // add the numbers; store result in sum
20
21    std::cout << "Sum is " << sum << std::endl; // display sum; end line
22 }
```

22

Now we understand how to output a message. Another fundamental concept is input. We need to take data from users so we can do some processing on the data. In C++, the input stream ...

(explain number1, number2,...)

Input - II

```
9 int number1; // first integer to add
10 int number2; // second integer to add
11 int sum; // sum of number1 and number2
```

- ❑ **Declarations** introduce the identifiers **number1**, **number2** and **sum** into programs; they are the names of **variables**.
- ❑ A variable is a location in the computer's memory where a value can be stored for use by a program.
- ❑ Variables **number1**, **number2** and **sum** are data of type **int**, meaning that these variables will hold **integer** values.
- ❑ All variables must be declared with a name and a data type before they can be used in a program.
- ❑ If more than one name is declared in a declaration, the names are separated by commas (,) → called **comma-separated list**.
 - ❑ `int number1,number2;`

23

Speaking of input, we must let the program know what data type your input is, right? Remember earlier in the slide I mention there are multiple data types, numbers, texts, binary, and so on. In our example, declaration ...

Input – III (other data types)

- ❑ Data type **double** is for specifying real numbers, and data type **char** for specifying character data.
 - ❑ Real numbers are numbers with decimal points, such as 3.4, 0.0 and -11.19.
- ❑ A **char** variable may hold only a single lowercase letter, a single uppercase letter, a single digit or a single special character (e.g., \$ or *).
- ❑ Types such as **int**, **double** and **char** are called **fundamental types**.
- ❑ Fundamental-type names are **keywords** and therefore must appear in all lowercase letters.

24

There are other data types defined by the C++ language. For example, ...

Fundamental Data Types

❑ <https://en.cppreference.com/w/cpp/language/types>

| Type specifier | Equivalent type | Width in bits by data model | | | | |
|------------------------|-----------------------------------|-----------------------------|------|-------|-------|------|
| | | C++ standard | LP32 | ILP32 | LLP64 | LP64 |
| short | short int | at least 16 | 16 | 16 | 16 | 16 |
| short int | | | | | | |
| signed short | | | | | | |
| signed short int | | | | | | |
| unsigned short | unsigned short int | | | | | |
| unsigned short int | | | | | | |
| int | int | at least 16 | 16 | 32 | 32 | 32 |
| signed | | | | | | |
| signed int | | | | | | |
| unsigned | | | | | | |
| unsigned int | unsigned int | | | | | |
| long | long int | at least 32 | 32 | 32 | 32 | 64 |
| long int | | | | | | |
| signed long | | | | | | |
| signed long int | | | | | | |
| unsigned long | unsigned long int | | | | | |
| unsigned long int | | | | | | |
| long long | long long int (C++11) | at least 64 | 64 | 64 | 64 | 64 |
| long long int | | | | | | |
| signed long long | | | | | | |
| signed long long int | | | | | | |
| unsigned long long | unsigned long long int (C++11) | | | | | |
| unsigned long long int | | | | | | |

25

This slides shows all the fundamental data types of C++. Each data type represents a particular number in integers or numerics. Each data type has a fixed width bit, where each bit represents a zero or one in the binary system of computer.

For example, if you define int, it has 32 bit; if you define double, it has 64 bit. The longer the bit width, the larger the value it can represent.

Range of Data Types

| Type | Size in bits | Format | Value range | |
|----------------|--------------|----------------------------|---|---|
| | | | Approximate | Exact |
| character | 8 | signed | | -128 to 127 |
| | | unsigned | | 0 to 255 |
| | 16 | unsigned | | 0 to 65535 |
| integer | 32 | unsigned | | 0 to 1114111 (0x10ffff) |
| | | signed | $\pm 3.27 \cdot 10^4$ | -32768 to 32767 |
| | 16 | unsigned | 0 to $6.55 \cdot 10^4$ | 0 to 65535 |
| | 32 | signed | $\pm 2.14 \cdot 10^9$ | -2,147,483,648 to 2,147,483,647 |
| | | unsigned | 0 to $4.29 \cdot 10^9$ | 0 to 4,294,967,295 |
| | 64 | signed | $\pm 9.22 \cdot 10^{18}$ | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| | | unsigned | 0 to $1.84 \cdot 10^{19}$ | 0 to 18,446,744,073,709,551,615 |
| floating point | 32 | IEEE-754 🔗 | <ul style="list-style-type: none"> min subnormal: $\pm 1.401,298,4 \cdot 10^{-45}$ min normal: $\pm 1.175,494,3 \cdot 10^{-38}$ max: $\pm 3.402,823,4 \cdot 10^{38}$ | <ul style="list-style-type: none"> min subnormal: $\pm 0x1p-149$ min normal: $\pm 0x1p-126$ max: $\pm 0x1.ffffep+127$ |
| | 64 | IEEE-754 🔗 | <ul style="list-style-type: none"> min subnormal: $\pm 4.940,656,458,412 \cdot 10^{-324}$ min normal: $\pm 2.225,073,858,507,201,4 \cdot 10^{-308}$ max: $\pm 1.797,693,134,862,315,7 \cdot 10^{308}$ | <ul style="list-style-type: none"> min subnormal: $\pm 0x1p-1074$ min normal: $\pm 0x1p-1022$ max: $\pm 0x1.ffffffffffffp+1023$ |

26

Here is the slide that summarizes the range of data types.

C++ is a Strongly Typed System

A key takeaway - C++ is a strongly typed system. All the variables you create must have a type. Each type has a predefined memory layout that the compiler can leverage to optimize the program.

Declare Variables

- ❑ Using the fundamental data types
 - ❑ e.g. `int my_variable`
- ❑ A variable name is any valid **identifier** that is not a keyword.
- ❑ An identifier is a series of characters consisting of letters, digits and underscores (`_`) that does **not begin with a digit**.
- ❑ C++ is **case sensitive**—uppercase and lowercase letters are different, so `a1` and `A1` are different identifiers.
- ❑ Declarations of variables can be placed almost anywhere in a program, but they must appear **before** their corresponding variables are used in the program.

28

Now we know the data type, we use it to declare variables.

Variable Naming is Important ...

- ❑ **Tip 1: C++ allows identifiers of any length, but your C++ implementation may restrict identifier lengths. Use identifiers of 31 characters or fewer to ensure portability**
- ❑ **Tip 2: choose meaningful identifiers makes a program self-explanatory—a person can understand the program simply by reading it rather than having to refer to the manuals or comments**
 - ❑ `int parent, PARENT;`
 - ❑ `int mom, dad;`
 - ❑ Good naming is always fundamental to effective code reviews and collaboration

Adding Integers - I

```
13 std::cout << "Enter first integer: "; // prompt user for data
14 std::cin >> number1; // read first integer from user into number1
```

- ❑ A **prompt** directs the user to take a specific action.
- ❑ A **cin** statement uses the **input stream object cin** (of namespace **std**) and the **stream extraction operator, >>**, to obtain a value from the keyboard.
- ❑ Using the stream extraction operator with **std::cin** takes character input from the standard input stream, which is usually the keyboard.

30

Let's go back to our previous example of input. I have declared a variable "number1" and now I want to assign it a value from the user. Here I use `std::cin`, which is a global object defined by the C++ library for your program to digest data from the console. When the program encounters this instruction, a prompt

Adding Integers - II

```
14  std::cin >> number1; // read first integer from user into number1
```

- ☐ When the computer executes an input statement, it waits for the user to enter a value for variable **number1**.
- ☐ The user types the number (as characters) then **press the Enter key** (or the Return key) to send the characters to the computer.
- ☐ The computer converts the character representation of the number to an integer and put the **value** to the variable **number1**.
- ☐ Any subsequent references to **number1** in this program will use this same value.

31

When the program execution encounter the `std::cin` instruction, it waits for the user to enter ...

Adding Integers - III

19

```
sum = number1 + number2; // add the numbers; store result in sum
```

- ❑ In this program, an assignment statement adds the values of variables **number1** and **number2** and assigns the result to variable **sum** using the **assignment operator =**.
 - ❑ Most calculations are performed in assignment statements.
- ❑ The **= operator** and the **+ operator** are called **binary operators** because each has two operands.

32

In this program ... using the assignment operator '='. Assignment operator is the most fundamental building block to construct an expression. A program consists of multiple expressions to describe your behavior.

Notice that the equal sign "=" has nothing to do with the equation you learn from high school, for example, you may need to solve an equation $y \text{ equals } x^2 \text{ plus } 1$. That is an equation. In C++ program, this is an assignment. The equal sign is an assignment operator and the + is a binary operator that adds two number and return the result to the left-hand side.

Adding Integers - IV

```
21      std::cout << "Sum is " << sum << std::endl; // display sum; end line
```

- ❑ **std::endl is a so-called stream manipulator.**
 - ❑ The name endl is an abbreviation for “end line”
- ❑ **std::endl outputs a newline and flushes the output buffer.**
 - ❑ Some systems may accumulate outputs in the machine (output buffer) and display them simultaneously on the screen.
 - ❑ std::endl forces any accumulated outputs to be displayed.
- ❑ **Using multiple stream insertion operators (<<) in a single statement is referred to as concatenating, chaining or cascading stream insertion operations.**
- ❑ **Calculations can also be performed in output statements.**

33

Once we complete this simple addition, we can cout the value of sum. Here I use std::endl to break the line. std::endl is a stream manipulator that outputs a new line and flushes the output stored in std::cout.

Memory Concepts

- ❑ Variable names such as **number1**, **number2** and **sum** actually correspond to **locations** in the computer's memory.
- ❑ Every variable has a **name**, a **type**, a **size** and a **value**.
- ❑ When a value is placed in a memory location, it **overwrites** the previous value in that location → **destructive**
- ❑ When a value is read out of a memory location, the process is **nondestructive**.

| | |
|---------|-----|
| number1 | 45 |
| number2 | 72 |
| sum | 117 |

34

Now we have seen a basic C++ program that takes two input number1 and number2 from the user, sum the two numbers, and store the result in a variable. It's important for us to know how this program actually works on hardware, or memory. Remember that in the first lecture, we talked about C++ is powerful in mapping your program directly into hardware. One of the fundamental concepts is memory architecture of your data.

(explain each item ...)

Yes, Memory is Limited

☐ You may not create too many variables

☐ e.g. one trillion integers or more



Surface Laptop 3 - 13.5", Black (metal), Intel Core i5, 8GB, 256GB

♡ Wish list

Slim and stylish, available in 13.5" and 15" touchscreens, rich color options,¹ and two durable finishes. Make a powerful statement and get improved speed, performance, and all-day battery life.²

 [Open gallery](#)

The Microsoft Store Promise for Surface

Shop with confidence at Microsoft Store. We're offering 60-day returns on Surface products, plus free digital workshops, remote learning opportunities, and more to help you get the most from your new device.*

[Learn more >](#)

Bundle and save with the Surface Laptop 3 Essentials Bundle

Includes your choice of Surface Laptop 3, Microsoft 365 and Microsoft Complete Protection Plan.

[Build your bundle >](#)

This literally means your program can run up to **8 GB** memory

35

You probably already know, memory is limited. Yes, that means you may not create too many variables, for example, a trillion integers or more. The largest size of the program you can run on your machine depends on the memory size you have. You can get this information when you purchase a laptop or a desktop. For example, below I am showing you a 13 inch surface laptop that has 8GB memory. This literally means your program can run up to 8 GB memory.

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output



Memory

| | |
|---------|----------|
| sum | 0012FF74 |
| | 0012FF75 |
| | 0012FF76 |
| | 0012FF77 |
| number2 | 0012FF78 |
| | 0012FF79 |
| | 0012FF7A |
| | 0012FF7B |
| number1 | 0012FF7C |
| | 0012FF7D |
| | 0012FF7E |
| | 0012FF7F |

To give you a better idea about how this simple program works on hardware, let's illustrate the runtime. Here I have the program that takes two inputs and sums their values to an output. On the right-hand side, I have the memory with variable name on the left and their virtual address from byte to byte on the right side. Remember each integer is indeed a 32-bit data, that is 4 bytes, because each byte has 8 bits.

Let's run the program.

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: _
```

Memory

| | |
|---------|----------|
| sum | 0012FF74 |
| | 0012FF75 |
| | 0012FF76 |
| | 0012FF77 |
| number2 | 0012FF78 |
| | 0012FF79 |
| | 0012FF7A |
| | 0012FF7B |
| number1 | 0012FF7C |
| | 0012FF7D |
| | 0012FF7E |
| | 0012FF7F |

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45_
```

Memory

| | |
|---------|----------|
| sum | 0012FF74 |
| | 0012FF75 |
| | 0012FF76 |
| | 0012FF77 |
| number2 | 0012FF78 |
| | 0012FF79 |
| | 0012FF7A |
| | 0012FF7B |
| number1 | 0012FF7C |
| | 0012FF7D |
| | 0012FF7E |
| | 0012FF7F |

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45
```

```
-
```

Memory

| | | |
|---------|----|----------|
| sum | | 0012FF74 |
| | | 0012FF75 |
| | | 0012FF76 |
| | | 0012FF77 |
| number2 | | 0012FF78 |
| | | 0012FF79 |
| | | 0012FF7A |
| | | 0012FF7B |
| number1 | 45 | 0012FF7C |
| | | 0012FF7D |
| | | 0012FF7E |
| | | 0012FF7F |

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45  
Enter second integer: _
```

Memory

| | | |
|---------|----|----------|
| sum | | 0012FF74 |
| | | 0012FF75 |
| | | 0012FF76 |
| | | 0012FF77 |
| number2 | | 0012FF78 |
| | | 0012FF79 |
| | | 0012FF7A |
| | | 0012FF7B |
| number1 | 45 | 0012FF7C |
| | | 0012FF7D |
| | | 0012FF7E |
| | | 0012FF7F |

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45  
Enter second integer: 72_
```

Memory

| | | |
|---------|----|----------|
| sum | | 0012FF74 |
| | | 0012FF75 |
| | | 0012FF76 |
| | | 0012FF77 |
| number2 | | 0012FF78 |
| | | 0012FF79 |
| | | 0012FF7A |
| | | 0012FF7B |
| number1 | 45 | 0012FF7C |
| | | 0012FF7D |
| | | 0012FF7E |
| | | 0012FF7F |

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45  
Enter second integer: 72  
—
```

Memory

| | | |
|---------|----|----------|
| sum | | 0012FF74 |
| | | 0012FF75 |
| | | 0012FF76 |
| | | 0012FF77 |
| number2 | 72 | 0012FF78 |
| | | 0012FF79 |
| | | 0012FF7A |
| | | 0012FF7B |
| number1 | 45 | 0012FF7C |
| | | 0012FF7D |
| | | 0012FF7E |
| | | 0012FF7F |

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45  
Enter second integer: 72  
—
```

Memory

| | | |
|---------|-----|----------|
| sum | 117 | 0012FF74 |
| | | 0012FF75 |
| | | 0012FF76 |
| | | 0012FF77 |
| number2 | 72 | 0012FF78 |
| | | 0012FF79 |
| | | 0012FF7A |
| | | 0012FF7B |
| number1 | 45 | 0012FF7C |
| | | 0012FF7D |
| | | 0012FF7E |
| | | 0012FF7F |

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45  
Enter second integer: 72  
Sum is 117_
```

Memory

| | | |
|---------|-----|----------|
| sum | 117 | 0012FF74 |
| | | 0012FF75 |
| | | 0012FF76 |
| | | 0012FF77 |
| number2 | 72 | 0012FF78 |
| | | 0012FF79 |
| | | 0012FF7A |
| | | 0012FF7B |
| number1 | 45 | 0012FF7C |
| | | 0012FF7D |
| | | 0012FF7E |
| | | 0012FF7F |

Illustration

```
std::cout << "Enter first integer: ";  
std::cin >> number1;  
std::cout << "Enter second integer: ";  
std::cin >> number2;  
sum = number1 + number2;  
std::cout << "Sum is " << sum << std::endl;
```

Output

```
Enter first integer: 45  
Enter second integer: 72  
Sum is 117  
Press any key to continue_
```

Memory

| | | |
|---------|-----|----------|
| sum | 117 | 0012FF74 |
| | | 0012FF75 |
| | | 0012FF76 |
| | | 0012FF77 |
| number2 | 72 | 0012FF78 |
| | | 0012FF79 |
| | | 0012FF7A |
| | | 0012FF7B |
| number1 | 45 | 0012FF7C |
| | | 0012FF7D |
| | | 0012FF7E |
| | | 0012FF7F |

| binary representation | hexadecimal | decimal |
|-------------------------------------|-------------|---------|
| 00000000 00000000 00000000 00000000 | 00000000 | 0 |
| 00000000 00000000 00000000 00000001 | 00000001 | 1 |
| 00000000 00000000 00000000 00000010 | 00000002 | 2 |
| 00000000 00000000 00000000 00000011 | 00000003 | 3 |
| 00000000 00000000 00000000 00000100 | 00000004 | 4 |
| 00000000 00000000 00000000 00000101 | 00000005 | 5 |
| 00000000 00000000 00000000 00000110 | 00000006 | 6 |
| 00000000 00000000 00000000 00000111 | 00000007 | 7 |
| 00000000 00000000 00000000 00001000 | 00000008 | 8 |
| 00000000 00000000 00000000 00001001 | 00000009 | 9 |
| 00000000 00000000 00000000 00001010 | 0000000A | 10 |
| 00000000 00000000 00000000 00001011 | 0000000B | 11 |
| 00000000 00000000 00000000 00001100 | 0000000C | 12 |
| 00000000 00000000 00000000 00001101 | 0000000D | 13 |
| 00000000 00000000 00000000 00001110 | 0000000E | 14 |
| 00000000 00000000 00000000 00001111 | 0000000F | 15 |

Value Representation

☐ Binary, Octal, Decimal, Hex

- ☐ The six letters (in addition to the 10 integers) in **hexadecimal** represent: 10 (A), 11 (B), 12 (C), 13 (D), 14 (E), and 15 (F), respectively.

| Name | Radix | Digits |
|-------------|-------|---------------------------------|
| Binary | 2 | 0,1 |
| Octal | 8 | 0,1,2,3,4,5,6,7 |
| Decimal | 10 | 0,1,2,3,4,5,6,7,8,9 |
| Hexadecimal | 16 | 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F |

Common Arithmetic Operators

| C++ operation | C++ arithmetic operator | Algebraic expression | C++ expression |
|----------------|-------------------------|--|--------------------|
| Addition | + | $f + 7$ | <code>f + 7</code> |
| Subtraction | - | $p - c$ | <code>p - c</code> |
| Multiplication | * | bm or $b \cdot m$ | <code>b * m</code> |
| Division | / | x / y or $\frac{x}{y}$ or $x \div y$ | <code>x / y</code> |
| Modulus | % | $r \bmod s$ | <code>r % s</code> |

48

We talked about the addition operation. Indeed, there are many more. In C++, we have subtraction that subtracts a number from another, multiplication that multiplies a number by another, division that divide a number by another, and modulus operation that finds the remainder of a division, for example, if you divide 7 by 3, the remainder is 1.

Summary

- ☐ **C++ input and output**
- ☐ **Variables, data types**
 - ☐ C++ is a strongly-typed system
- ☐ **Adding two integers**
- ☐ **Value representation (binary, hex, decimal)**
- ☐ **Arithmetic operators (+, -, *, /)**

LAB 1

- ☐ Write a program that declare three floating numbers **a=2, b=3, and c=4**
 - ☐ Store the summation of a, b, c in variable x
 - ☐ What is the value of `std::cout << b/a`?
 - ☐ What is the value of `std::cout << b/c`?
 - ☐ How about changing the type from float to int?