

Tutorial 2

Get familiar with VS Code & Makefile and String Applications

Sep. 19, 2022

Lai Wei (USTF)

(SDS, 120090485@link.cuhk.edu.cn)

Objectives today

1. Some background information about this course.
2. Two concepts clarification
3. Get familiar with some common operations of string (e.g., substr), **which will be important in your Assignment 1!**
4. Three examples with string operations:
 1. Palindromes 回文数
 2. Acronym 首字母缩略词
 3. Pig Latin 儿童隐语
5. Learn to use VS Code and Makefile to run the examples above.
6. Q&A time: make sure you can run the C++ code on your computer now! (If you can't, solve it today!)

1. Background

- The course changed a lot in this semester. Now, you can:
- Use Qt Creator to write code. (Recommended)
 - Most examples in our textbook has implementations with the Stanford Library.
 - It is difficult to embed the Library into other IDEs and there is a blank or empty project provided with the Library already installed.
 - So if you want to use Stanford Library, you probably need to use Qt.
- Use VS Code to write code. (Highly Recommended)
 - Familiar, Light-weight, easy to use.
 - Learn to use command lines and Makefile.
 - Allow writing and running single .cpp file without create a heavy “project”.
 - Our assignments are individual problems (individual files), VS Code is enough to use.
- Use other IDEs like Visual Studio or CLion. (Optional)

1. Background

- A fact:
- Our commonly used “Python” is mainly implemented in C language. (<https://github.com/python/cpython>)
- C/C++ language is more complicated than Python. Python hides all complicated grammar and details in C/C++, providing you an easy-to-use programming language.
- **Why more complicated? Designed for high performance.**
- But in this course, we are learning C++. It is common for you to feel more difficult than CSC1001 (Python). But don't worry!

2. Concepts

- What is g++?
 - g++ is the traditional nickname of GNU C++, a freely redistributable **C++ compiler** produced by the Free Software Foundation plus dozens of skilled volunteers
 - *(source: http://tin2.vub.ac.be/~dvermeir/manual/g++/g++faq_1.html)*
- What is Makefile?
 - **Makefile (Script)** is a way of **automating software building procedure** with dependencies.
 - The **make (program)** automatically determines which pieces of a large program need to be recompiled, and issues commands to recompile them.
 - **NOTE: make (program)** is only a helper building tool, not the compiler. The core compiler is still g++.
 - *(source: <https://www3.nd.edu/~zxu2/acms60212-40212/Makefile.pdf>)*

3. Get familiar with some common operations of string

- Two types of String:

- C-style string (char array)

- `char c[6] = {'h', 'e', 'l', 'l', 'o', '\0'};`
- `char c[6] = "hello";`

Last one be '\0'
to make it
safe!!

- C++ `std::string`

- `include<string>`
- `std::string s = "hello";`

length (6-1)
maximum!!!

- Changing C-style string to `std::string`

- `std::string s(c);`
- `std::string s = std::string(c);`

- Changing `std::string` to C-style string

- `s.c_str();`

3. The <cctype> (ctype.h) Interface

This header declares a set of functions to **classify** and **transform individual characters**.

<code>bool isdigit(char ch)</code> Determines if the specified character is a digit.
<code>bool isalpha(char ch)</code> Determines if the specified character is a letter.
<code>bool isalnum(char ch)</code> Determines if the specified character is a letter or a digit.
<code>bool islower(char ch)</code> Determines if the specified character is a lowercase letter.
<code>bool isupper(char ch)</code> Determines if the specified character is an uppercase letter.
<code>bool isspace(char ch)</code> Determines if the specified character is <i>whitespace</i> (spaces and tabs).
<code>char tolower(char ch)</code> Converts <code>ch</code> to its lowercase equivalent, if any. If not, <code>ch</code> is returned
<code>char toupper(char ch)</code> Converts <code>ch</code> to its uppercase equivalent, if any. If not, <code>ch</code> is returned unchanged.

For more, please visit <https://cplusplus.com/reference/cctype/>

3. The <cstring> (string.h) Interface

This header file defines several functions to **manipulate C strings** and **arrays**.

<pre>void* memcpy(void* dst, const void* src, size_t num)</pre> <p>Copy block of memory</p>
<pre>char* strcat(char* dst, const char* src)</pre> <p>Concatenate strings</p>
<pre>void* memchr (void* ptr, int value, size_t num)</pre> <p>Locate character in block of memory</p>
<p>.....</p>

For more, please visit <https://cplusplus.com/reference/cstring/>

3. Operators on the string Class

- To convert the C++ string objects into C string literals, simply apply the `c_str` method to the C++ string.
- Unlike most languages, C++ allows classes to redefine the meanings of the standard operators. As a result, several string operations, such as `+` for concatenation, are implemented as operators (**overloading**).

<code>str[i]</code> Returns the i^{th} character of <code>str</code> . Assigning to <code>str[i]</code> changes that character.
<code>s1 + s2</code> Returns a new string consisting of <code>s1</code> concatenated with <code>s2</code> .
<code>s1 = s2;</code> Copies the character string <code>s2</code> into <code>s1</code> .
<code>s1 += s2;</code> Appends <code>s2</code> to the end of <code>s1</code> .
<code>s1 == s2</code> (and similarly for <code><</code> , <code><=</code> , <code>></code> , <code>>=</code> , and <code>!=</code>) Compares to strings lexicographically.
<code>str.c_str()</code> Returns a C-style character array containing the same characters as <code>str</code> .

3. Operators on the string Class

`str.length()`

Returns the number of characters in the string `str`.

`str.at(index)`

Returns the character at position `index`; most clients use `str[index]` instead.

`str.substr(pos, len)`

Returns the substring of `str` starting at `pos` and continuing for `len` characters.

`str.find(ch, pos)`

Returns the first index \geq `pos` containing `ch`, or `string::npos` if not found.

`str.find(text, pos)`

Similar to the previous method, but with a string instead of a character.

4.1. Palindrome

- A *palindrome* is a word that reads identically backward and forward, such as “level” or “noon”.
- Write a C++ program `isPalindrome` that checks whether a string is a palindrome.

```
bool isPalindrome(string str) {  
    int n = str.length();  
    for (int i = 0; i < n / 2; i++) {  
        if (str[i] != str[n - i - 1]) return false;  
    }  
    return true;  
}
```

```
bool isPalindrome(string str) {  
    return str == reverse(str);  
}
```

we define this
function to reverse a
string.

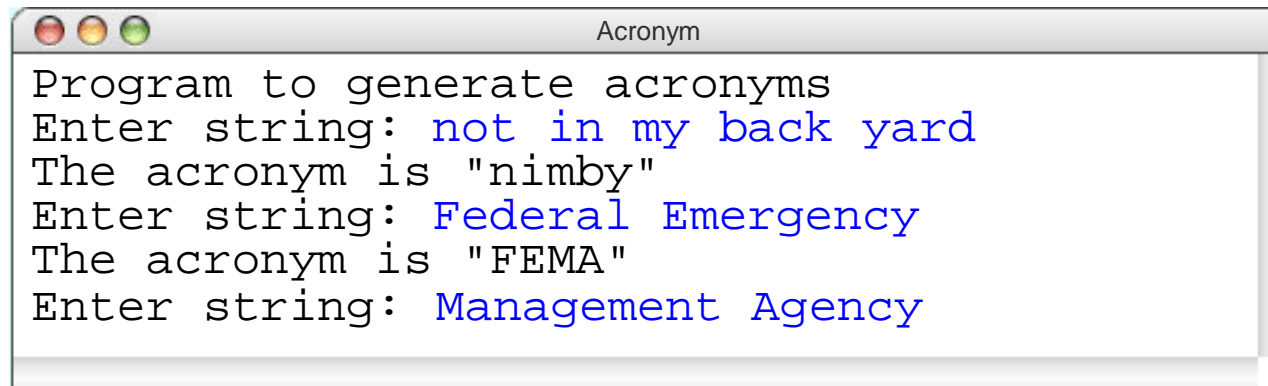
- Efficiency vs. Readability

4.2. Acronym

- An *acronym* is a word formed by taking the first letter of each word in a sequence, as in

"self-contained underwater breathing apparatus" → "scuba"

- Write a C++ program that generates acronyms, as illustrated by the following sample run:



```
Acronym
Program to generate acronyms
Enter string: not in my back yard
The acronym is "nimby"
Enter string: Federal Emergency
The acronym is "FEMA"
Enter string: Management Agency
```

4.2. Acronym

```
string acronym(string str) {
    string result = "";
    bool inWord = false;
    int nc = str.length();
    for (int i = 0; i < nc; i++) {
        char ch = str[i];
        if (inWord) {
            if (!isalpha(ch)) inWord = false;
        } else {
            if (isalpha(ch)) {
                result += ch;
                inWord = true;
            }
        }
    }
    return result;
}
```

4.3. Translating English to Pig Latin

We describe a C++ program that reads a line of text from the user and then translates each word in that line from English to Pig Latin, a made-up language familiar to most children in the English-speaking world.

In Pig Latin, words are formed from their English counterparts by applying the following rules:

1. If the word contains no vowels (元音), no translation is done, which means that the translated word is the same as the original.
2. If the word begins with a vowel, the function adds the string "way" to the end of the original word.
3. If the word begins with a consonant (辅音), the function extracts the string of consonants up to the first vowel, moves that collection of consonants to the end of the word, and adds the string "ay".

4.3. Translating English to Pig Latin

As an example, suppose that the English word is *scram*. **Because the word begins with a consonant**, you divide it into two parts: one consisting of the letters before the first vowel and one consisting of that vowel and the remaining letters:

s c r **a m**

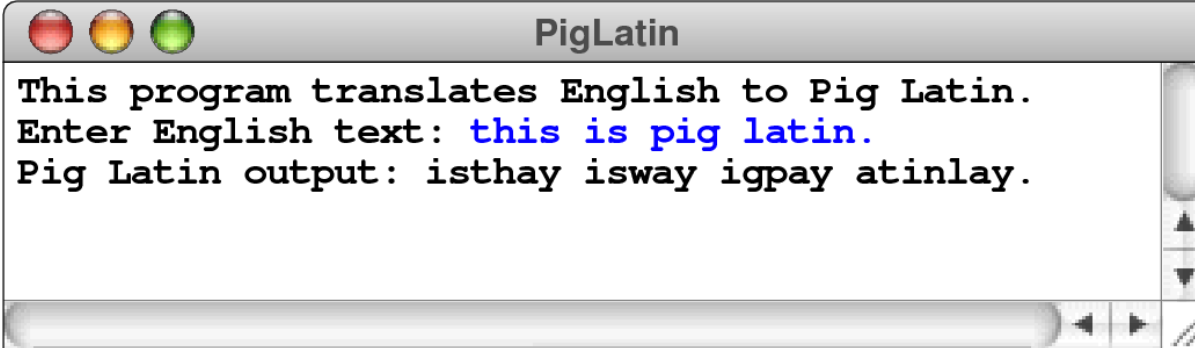
You then interchange these two parts and add *ay* at the end, as follows:

a m **s c r** **a y**

Thus the Pig Latin word for *scram* is *amscray*. **For a word that begins with a vowel**, such as *apple*, you simply add *way* to the end, which leaves you with *appleway*.

4.3. Translating English to Pig Latin

A sample run of the program might look like this:



```
PigLatin
This program translates English to Pig Latin.
Enter English text: this is pig latin.
Pig Latin output: isthay isway igpay atinlay.
```

It is worth taking a careful look at the implementations of *lineToPigLatin* and *wordToPigLatin*. The *lineToPigLatin* function finds the word boundaries in the input and provides a useful pattern for separating a string into individual words. The *wordToPigLatin* function uses *substr* to extract pieces of the English word and then uses concatenation to put them back together in their Pig Latin form. In Chapter 6, you will learn about a more general facility called a token scanner that divides a string into its logically connected parts.

5.1. Run the code via command lines (terminal)

- If you don't know how, or fail to set the VS Code makefile extension (settings.json), you can **ALWAYS** use **COMMAND LINES** to compile and run your C++ code.
- Only pre-requisite: you can run “make --version” and “g++ --version” in the command lines.

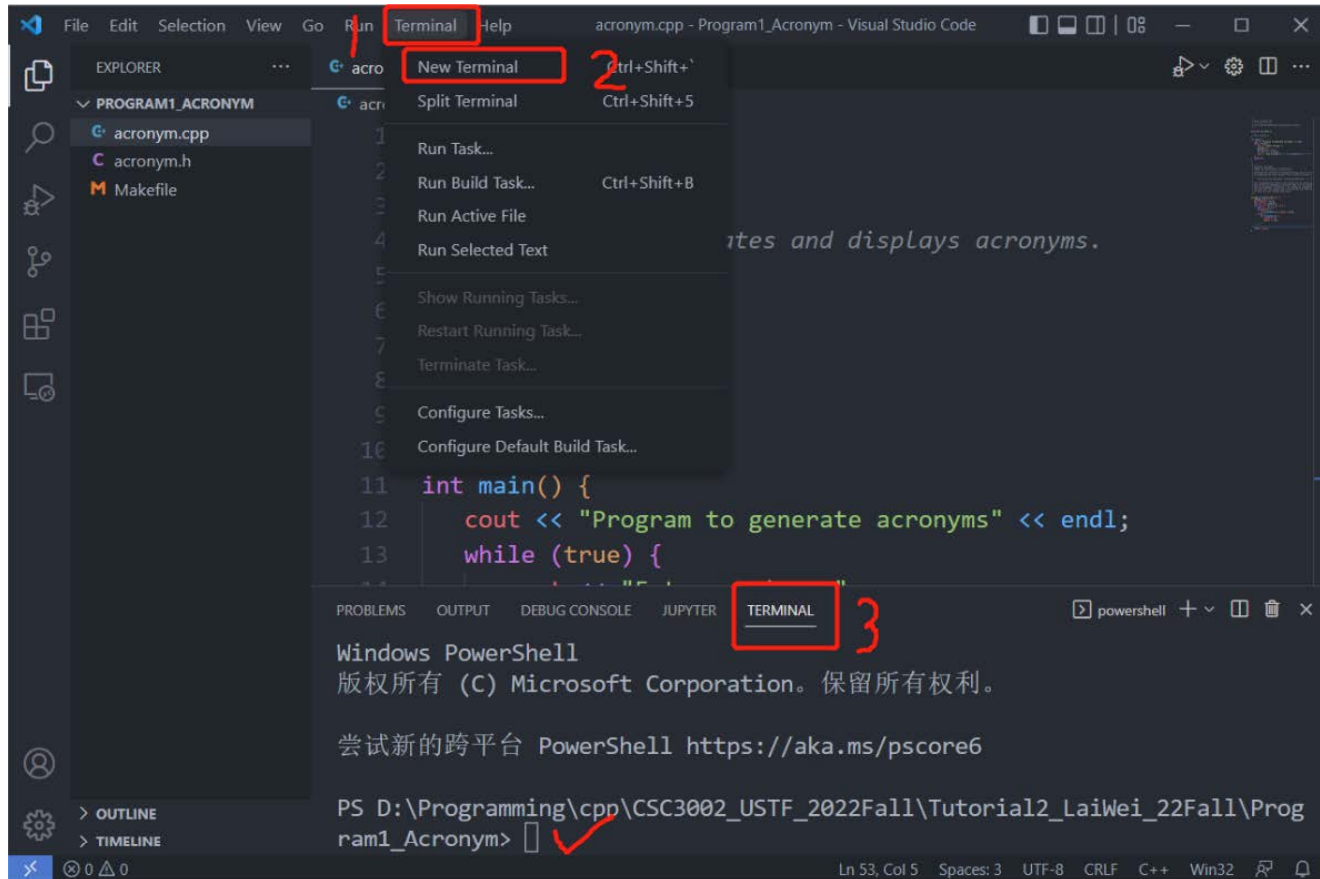
```
PS C:\Users\30785> make --version
GNU Make 3.81
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

This program built for i386-pc-mingw32
PS C:\Users\30785> g++ --version
g++.exe (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 7.3.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

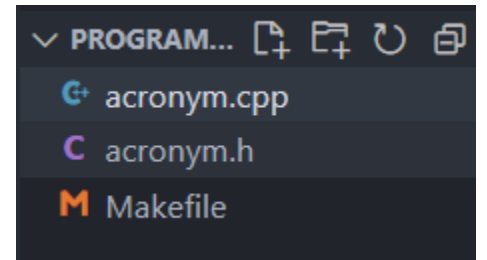
5.1. Run the code via command lines

a) Compile by **pure** command lines, with “g++”
compile command

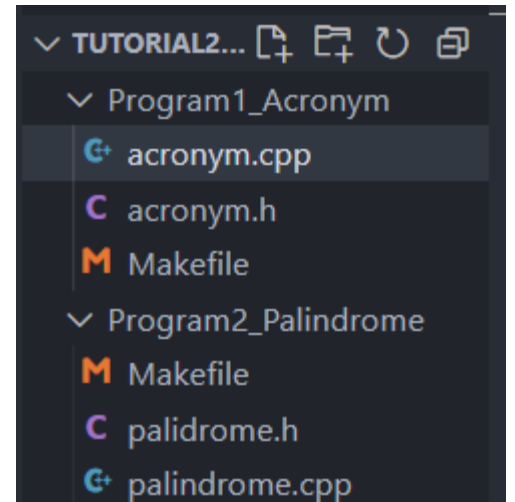
- Open a terminal in the current code folder.



Correct: Makefile is in current program folder workspace, no path problem.



Wrong: Makefile in NOT in program folder workspace, may bring relative-path problem.



5.1. Run the code via command lines

a) Compile by **pure** command lines, with “g++”
compile command

- In the terminal, type compile command:
- “g++ -std=c++17 <your source .cpp files> -o <output filename>”
- E.g. “g++ -std=c++17 helloworld.cpp foo.cpp -o helloworld”

- Then run the executable program:
- different terminal have different calling method, maybe:
 - “./<filename>”, “<filename>” (macOS system)
 - “./<filename>.exe”, “<filename>.exe” (Windows system)

5.1. Run the code via command lines

a) Compile by **pure** command lines, with “g++”
compile command

尝试新的跨平台 PowerShell <https://aka.ms/pscore6>

```
PS D:\Programming\cpp\CSC3002_USTF_2022Fall\Tutorial2_LaiWei_22Fall\Program1_Acronym> g++ -std=c++17 acronym.  
cpp -o acronym.exe
```

```
PS D:\Programming\cpp\CSC3002_USTF_2022Fall\Tutorial2_LaiWei_22Fall\Program1_Acronym> ./acronym.exe
```

```
Program to generate acronyms
```

```
Enter string: █
```

compile

run

- Use <ctrl>(command) + C to exit the program.

5.1. Run the code via command lines

b) Compile by **pure** command lines, with “makefile”.

- Write your own “Makefile” script, or use the template given. (If you use Prof Kinley’s template, remember to change two names!)
- In the terminal, type make command:
- “make” (or “mingw32-make.exe”, if you use Qt’s make tool)
- Then run the executable program:
- different terminal have different calling method, maybe:
 - “./<filename>”, “<filename>” (macOS system)
 - “./<filename>.exe”, “<filename>.exe” (Windows system)

```
45 PROGRAM = \  
46     helloworld  
47  
48 OBJECTS = \  
49     helloworld.o \  
50     foo.o \  
51
```

5.1. Run the code via command lines

b) Compile by **pure** command lines, with “makefile”.

尝试新的跨平台 PowerShell <https://aka.ms/pscore6>

```
PS D:\Programming\cpp\CSC3002_USTF_2022Fall\Tutorial2_LaiWei_22Fall\Program1_Acronym> make
g++ -c -std=c++17 acronym.cpp -o acronym.o
g++ -std=c++17 acronym.o -o acronym run
PS D:\Programming\cpp\CSC3002_USTF_2022Fall\Tutorial2_LaiWei_22Fall\Program1_Acronym> ./acronym.exe
Program to generate acronyms
Enter string: █
```

- Use <ctrl>(command) + C to exit the program.

5.1. Run the code via command lines

b) Compile by **pure** command lines, with “makefile”.

- One useful tip: Use “↑” “↓”(Up/down arrows on the keyboard) to view command history.
- Two useful tips: Use “Tab” for Automatic completion the command / filename
- (Demo)

5.2. Run the code via VS Code make-extension

- A fact behind the make-extension: **it automatically generates commands in terminal** for you after you **pressing the “build” “run” button**. In the first time you press the button, it automatically generates configurations for you (in “.vscode/settings.json”).
- If it succussed, you can use the button to compile and run your program.
- But if it failed, you need to manually set the proper path settings (in “.vscode/settings.json”).
- But, you can **ALWAYS** use **COMMAND LINES** to compile and run your C++ code.

5.2. Run the code via VS Code make-extension

- You can **first directly use the buttons** to compile and run your program without setting path in “settings.json”. The Extension will **automatically detect and generate the settings** for you. If you find the setting path is not correct, view next-next slide for solution.

- Build:

```
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000
```

```
all: $(PROGRAM)

$(PROGRAM): $(OBJECTS)
    $(CXX) $(CXXFLAGS) $(OBJECTS) -o $@
```

```
Configure elapsed time: 0.106
The build targets list may not be accurate because configure failed.
```

5.2. Run the code via VS Code make-extension

- Run:

The screenshot shows the Visual Studio Code interface with a Makefile configuration error. The error message in the top right says "No launch targets identified". A red box highlights this message, and a red arrow points to the terminal output below. The terminal output shows the command `helloworld.exe()` being executed. A yellow box highlights the `launchConfigurations` section of the `settings.json` file, which contains the following configuration:

```
launchConfigurations": [  
  {  
    "name": "D:\\Programming\\cpp\\CSC3002_USTF_2022Fall\\helloworld\\",  
    "type": "cppdbg",  
    "request": "launch",  
    "program": "D:\\Programming\\cpp\\CSC3002_USTF_2022Fall\\helloworld\\helloworld.exe",  
    "args": []  
  }  
]
```

A yellow arrow points from the error message to the `program` field in the configuration, with the text "correspond to this" next to it. The terminal output at the bottom shows the following error message:

```
Configure elapsed time: 0.089  
The launch targets list may not be accurate because configure failed.
```

5.2. Run the code via VS Code make-extension

- If you find the auto-generated setting path is not correct, you can manually set the path in the “settings.json”:

The screenshot shows the Visual Studio Code interface with the `settings.json` file open. The file content is as follows:

```
1 {
2   "makefile.launchConfigurations": [
3     {
4       "cwd": "D:\\Programming\\cpp\\CSC3002_USTF_2022Fall\\helloworld\\",
5       "binaryPath": "D:\\Programming\\cpp\\CSC3002_USTF_2022Fall\\helloworld\\helloworld.exe",
6       "binaryArgs": []
7     },
8   ],
9   "C_Cpp.default.configurationProvider": "ms-vscode.makefile-tools",
10  "C_Cpp.default.compilerPath": "ms-vscode.makefile-tools",
11  "C_Cpp.default.compilerQuiggles": "Disabled",
12  "C_Cpp.default.compilerAssociations": {
13    "stream": "cpp"
14  }
15 }
```

Annotations and instructions:

- 1** double-click the blank area, Then right click the blank area
- 2** Click this to copy the absolute path to this folder
- 3** Paste the folder path to here
- 4** Paste the folder path here, add with your output filename

The `Copy Path` option in the context menu is highlighted with a red box.

6. Q & A time

- Thank you for your listening!
- Lai Wei (USTF)
- (SDS, 120090485@link.cuhk.edu.cn)
- Office hour: Friday 10:00-11:00 am, Start-up Zone library L103