```
1 // Created by hengxin on 11/06/24.
2 // pythontutor:
3 // https://pythontutor.com/render.html#code=%23include%20%
  3Cstdio.h%3E%0A%0A%23define%20PI%203.14%0A%0Aint%20main%
  28void%29%20%7B%0A%20%20//%20https%3A//intellij-support.
  jetbrains.com/hc/en-us/community/posts/115000740490-Where-
  did-the-black-windows-go-%3Fpage%3D1%
  23community_comment_115000619510%0A%20%20//%20%20setbuf%
  28stdout,%20NULL%29%3B%0A%0A%20%20/**********%20On%
  20radius_1%20**********/%0A%20%20//%20type%3A%20int%3B%
  20value%3A%20100%3B%20address%3A%20%26radius_1%20%28%26%3A
  %20address-of%20operator%29%0A%20%20int%20radius_1%20%3D%
  20100%3B%0A%0A%20%20printf%28%22radius_1%20%3D%20%25d%5Cn%
  22,%20radius_1%29%3B%0A%20%20printf%28%22%26radius_1%20%3D
  %20%25p%5Cn%22,%20%26radius_1%29%3B%0A%0A%20%20//%20lvalue
  %20and%20rvalue%0A%20%20radius_1%20%3D%20200%3B%0A%20%20
  //%20lvalue%20conversion%20%3D%3E%20rvalue%0A%20%20double%
  20circumference%20%3D%202%20*%20PI%20*%20radius_1%3B%0A%20
  %20printf%28%22circumference%20%3D%20%25f%5Cn%22,%
  20circumference%29%3B%0A%20%20/**********%20On%20radius_1%
  20**********/%0A%0A%20%20/**********%20On%20ptr_radius_1%
  20**********/%0A%20%20//%20type%3A%20int%20*%3B%20value%3A
  %20%26radius_1%0A%20%20int%20*ptr_radius_1%20%3D%20%
  26radius_1%3B%0A%0A%20%20printf%28%22%26ptr_radius_1%20%3D
  %20%25p%5Cn%22,%20%26ptr_radius_1%29%3B%0A%20%20
  /**********%20On%20ptr_radius_1%20**********/%0A%0A%20%20
  /**********%20On%20ptr_radius_1%20as%20lvalue%20and%
  20rvalue%20**********/%0A%20%20int%20radius_2%20%3D%201000
  %3B%0A%20%20int%20*ptr_radius_2%20%3D%20%26radius_2%3B%0A%
  0A%20%20ptr_radius_1%20%3D%20ptr_radius_2%3B%0A%20%20*
  ptr_radius_2%20%3D%202000%3B%0A%20%20printf%28%22radius_1%
  20%3D%20%25d%5Cn%22,%20*ptr_radius_1%29%3B%0A%20%20
  /**********%20On%20ptr_radius_1%20as%20lvalue%20and%
  20rvalue%20**********/%0A%0A%20%20/**********%20On%20*
  ptr_radius_1%20**********/%0A%20%20//%20*%3A%20indirection
  /dereference%20operator%0A%20%20//%20*ptr_radius_1%
  20behaves%20like%20radius_1%0A%20%20*ptr_radius_1%20%3D%
  20200%3B%0A%20%20printf%28%22radius_1%20%3D%20%25d%5Cn%22
  ,%20radius_1%29%3B%0A%20%20circumference%20%3D%202%20*%
  20PI%20*%20%28*ptr_radius_1%29%3B%0A%20%20/**********%20On
  %20*ptr_radius_1%20**********/%0A%0A%20%20int%20v%20%3D%
  20100%3B%0A%20%20int%20*pv%20%3D%20%26v%3B%0A%20%20printf%
  28%22pv%20%3A%20%25p%5Cn%20*pv%20%3A%20%25d%5Cn%22,%20pv,%
  20*pv%29%3B%0A%20%20pv%20%3D%20%26pv%3B%0A%20%20printf%28%
```

```
 3 22pv%20%3A%20%25p%5Cn%22,%20pv%29%3B%0A%0A%20%20return%200
   %3B%0A%7D&cumulative=true&curInstr=0&heapPrimitives=
   nevernest&mode=display&origin=opt-frontend.js&py=c_gcc9.3.
   0&rawInputLstJSON=%5B%5D&textReferences=false
 4
 5 #include <stdio.h>
 6
 7 #define PI 3.14
 8
 9 int main(void) {
10   // https://intellij-support.jetbrains.com/hc/en-us/
   community/posts/115000740490-Where-did-the-black-windows-
   go-?page=1#community_comment_115000619510
11   //  setbuf(stdout, NULL);
12
13   /********** On radius_1 **********/
14   // type: int; value: 100; address: &radius_1 (&: address
   -of operator)
15   int radius_1 = 100;
16
17   printf("radius_1 = %d\n", radius_1);
18   printf("&radius_1 = %p\n", &radius_1);
19
20   // lvalue and rvalue
21   radius_1 = 200;
22   // lvalue conversion => rvalue
23   double circumference = 2 * PI * radius_1;
24   printf("circumference = %f\n", circumference);
25   /********** On radius_1 **********/
26
27   /********** On ptr_radius_1 **********/
28   // type: int *; value: &radius_1
29   int *ptr_radius_1 = &radius_1;
30
31   printf("&ptr_radius_1 = %p\n", &ptr_radius_1);
32   /********** On ptr_radius_1 **********/
33
34   /********** On ptr_radius_1 as lvalue and rvalue
    **********/
35   int radius_2 = 1000;
36   int *ptr_radius_2 = &radius_2;
37
38   ptr_radius_1 = ptr_radius_2;
39   *ptr_radius_2 = 2000;
```

```c
    printf("radius_1 = %d\n", *ptr_radius_1);
    /********** On ptr_radius_1 as lvalue and rvalue
    **********/

    /********** On *ptr_radius_1 **********/
    // *: indirection/dereference operator
    // *ptr_radius_1 behaves like radius_1
    *ptr_radius_1 = 200;
    printf("radius_1 = %d\n", radius_1);
    circumference = 2 * PI * (*ptr_radius_1);
    /********** On *ptr_radius_1 **********/

    int v = 100;
    int *pv = &v;
    printf("pv : %p\n *pv : %d\n", pv, *pv);
    pv = &pv;
    printf("pv : %p\n", pv);

    return 0;
}
```

```
 1 # 8-pointer
 2
 3 ## `radius.c`
 4
 5 ### On Variables
 6 - type, value, address
 7 - `&`: address-of operator
 8 - printf the address (`%p`)
 9 - `lvalue`, `rvalue`???
10
11 ### On Pointers
12 - `int *` syntax
13 - int * vs. double * (type cast???)
14 - refs to itself (int ** vs. int *)
15 - Visualization
16
17 - `scanf`: how does it work???
18
19 ## `Swap` (`selection-sort.c`)
20 - `WrongSwap`
21 - `Swap`
22 - Visualization
23
24 ## Pointers and Arrays (`selection-sort.c`)
25
26 - `()`: function call operator
27   - `SelectionSort(numbers, LEN)`
28 - `int arr[]` vs. `(int *arr)`
29 - `numbers[i]` vs. `*(numbers + i)`
30   - pointers arithmetic (in arrays!!!)
31   - `pointer + int`, `pointer - int`, `pointer - pointer`
32 - `&numbers[i]` vs. `numbers + i`
33
34 ## Array Name (`selection-sort.c`)
35 - `int arr[] = {1, 2, 3};`
36 - `arr++`
37 - `numbers++`
38
39 ## Dynamic Memory Management (`selection-sort.c`)
40
41 - VLA
42 - `malloc.h` vs. `stdlib.h`
43 - `malloc`
44   - `void *`
```

```
45      - `int *`
46      - `sizeof(*numbers)`
47    - size = 0: implementation-defined
48    - `unsigned long long`
49  - `NULL`
50    - `(void *) 0`
51  - `free`
52    - memory leak (heap)
53    - **undefined behaviors**
54      - double `free`
55      - `free` non-`malloc`
56        - `numbers = NULL`
57      - dereference `free`d memory
58
59  ## `const` in `Print` (`selection-sort.c`)
```

File - D:\cpl\2024-cpl-coding\7-pointers-arrays\CMakeLists.txt

```
1 add_executable(pointer pointer.c)
2 add_executable(selection-sort-pointers selection-sort.c)
3 add_executable(pointer-array pointer-array.c)
4 add_executable(pointer-const pointer-const.c)
```

```c
// Created by hfwei on 2024/11/6.

int main(void) {
  int arr[] = {1, 2, 3};
  // arr = arr + 1;

  /********** On malloc/free **********/
  int var = 10;
  // free(var);
  //  free(arr);
  /********** On malloc/free **********/

  return 0;
}
```

```c
// Created by hfwei on 2024/11/6.
// Python tutor:
// https://pythontutor.com/render.html#code=%23include%20%3Cstdio.h%3E%0A%0Aint%20main%28void%29%20%7B%0A%20%20//%20v%3A%20int,%20const%20int%0A%20%20//%20pv%3A%20int%20*,%20int%20*%20const%0A%20%20//%20const%20int%20*,%20const%20int%20*%20const,%20int%20const%20*,%20int%20const%20*%20const%0A%0A%20%20int%20var%20%3D%200%3B%0A%0A%20%20//%20int%20var_1%0A%20%20int%20var_1%20%3D%2010%3B%0A%0A%20%20int%20*ptr_1%20%3D%20%26var_1%3B%0A%20%20*ptr_1%20%3D%2020%3B%0A%20%20printf%28%22var_1%20%3D%20%25d%5Cn%22,%20var_1%29%3B%0A%20%20ptr_1%20%3D%20%26var%3B%0A%0A%20%20const%20int%20*ptr_1_1%20%3D%20%26var_1%3B%0A%20%20//%20%20*ptr_1_1%20%3D%2030%3B%20%20//%20Wrong%0A%20%20printf%28%22var_1%20%3D%20%25d%5Cn%22,%20var_1%29%3B%0A%20%20ptr_1%20%3D%20%26var%3B%0A%0A%20%20int%20*const%20ptr_1_2%20%3D%20%26var_1%3B%0A%20%20*ptr_1_2%20%3D%2040%3B%0A%20%20printf%28%22var_1%20%3D%20%25d%5Cn%22,%20var_1%29%3B%0A%20%20//%20%20*ptr_1_2%20%3D%20%26var%3B%20%20//%20Wrong%0A%0A%20%20//%20const%20int%20var_2%0A%20%20const%20int%20var_2%20%3D%20100%3B%0A%0A%20%20int%20*ptr_2%20%3D%20%26var_2%3B%0A%20%20*ptr_2%20%3D%20200%3B%0A%20%20printf%28%22var_2%20%3D%20%25d%5Cn%22,%20var_2%29%3B%0A%0A%20%20const%20int%20*ptr_2_1%20%3D%20%26var_2%3B%0A%20%20//%20%20*ptr_2_1%20%3D%20300%3B%0A%20%20printf%28%22var_2%20%3D%20%25d%5Cn%22,%20var_2%29%3B%0A%20%20const%20int%20*const%20ptr_2_2%20%3D%20%26var_2%3B%0A%20%20//%20%20*ptr_2_2%20%3D%20400%3B%0A%20%20printf%28%22var_2%20%3D%20%25d%5Cn%22,%20var_2%29%3B%0A%0A%20%20return%200%3B%0A%7D&cumulative=true&curInstr=0&heapPrimitives=nevernest&mode=display&origin=opt-frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D&textReferences=false

#include <stdio.h>

int main(void) {
  // v: int, const int
  // pv: int *, int * const
  // const int *, const int * const, int const *, int const * const

  int var = 0;

  // int var_1
```

```c
    int var_1 = 10;

    int *ptr_1 = &var_1;
    *ptr_1 = 20;
    printf("var_1 = %d\n", var_1);
    ptr_1 = &var;

    const int *ptr_1_1 = &var_1;
    //  *ptr_1_1 = 30;  // Wrong
    printf("var_1 = %d\n", var_1);
    ptr_1 = &var;

    int *const ptr_1_2 = &var_1;
    *ptr_1_2 = 40;
    printf("var_1 = %d\n", var_1);
    //  *ptr_1_2 = &var;  // Wrong

    // const int var_2
    const int var_2 = 100;

    int *ptr_2 = &var_2;
    *ptr_2 = 200;
    printf("var_2 = %d\n", var_2);

    const int *ptr_2_1 = &var_2;
    //  *ptr_2_1 = 300;
    printf("var_2 = %d\n", var_2);
    const int *const ptr_2_2 = &var_2;
    //  *ptr_2_2 = 400;
    printf("var_2 = %d\n", var_2);

    return 0;
}
```

```
1 // Created by hfwei on 2024/11/06.
2 // Visualization of Swap:
3 // https://pythontutor.com/render.html#code=//%20Created%
  20by%20hfwei%20on%202024/10/12.%0A%0A%0A%23include%20%
  3Cstdio.h%3E%0A%23include%20%3Cstdlib.h%3E%0A%0A%23define%
  20LEN%205%0A%0Avoid%20SelectionSort%28int%20arr%5B%5D,%
  20int%20len%29%3B%0Avoid%20WrongSwap%28int%20left,%20int%
  20right%29%3B%0Avoid%20Swap%28int%20*left,%20int%20*right%
  29%3B%0Aint%20GetMinIndex%28const%20int%20arr%5B%5D,%20int
  %20begin,%20int%20end%29%3B%0Avoid%20Print%28const%20int%
  20arr%5B%5D,%20int%20len%29%3B%0A%0Aint%20main%28void%29%
  20%7B%0A%20%20int%20numbers%5BLEN%5D%20%3D%20%7B%2025,%
  2078,%2015,%2023,%2011%20%7D%3B%0A%0A%20%20Print%28numbers
  ,%20LEN%29%3B%0A%20%20SelectionSort%28numbers,%20LEN%29%3B
  %0A%20%20Print%28numbers,%20LEN%29%3B%0A%0A%20%20return%
  200%3B%0A%7D%0A%0Avoid%20SelectionSort%28int%20*arr,%20int
  %20len%29%20%7B%0A%20%20for%20%28int%20i%20%3D%200%3B%20i%
  20%3C%20len%3B%20i%2B%2B%29%20%7B%0A%20%20%20%20int%
  20min_index%20%3D%20GetMinIndex%28arr,%20i,%20len%29%3B%0A
  %20%20%20%20Swap%28arr%20%2B%20i,%20arr%20%2B%20min_index%
  29%3B%0A%20%20%7D%0A%7D%0A%0Aint%20GetMinIndex%28const%
  20int%20*arr,%20int%20begin,%20int%20end%29%20%7B%0A%20%
  20int%20min%20%3D%20arr%5Bbegin%5D%3B%0A%20%20int%
  20min_index%20%3D%20begin%3B%0A%0A%20%20for%20%28int%20i%
  20%3D%20begin%20%2B%201%3B%20i%20%3C%20end%3B%20%2B%2Bi%29
  %20%7B%0A%20%20%20%20if%20%28arr%5Bi%5D%20%3C%20min%29%20%
  7B%0A%20%20%20%20%20%20min%20%3D%20arr%5Bi%5D%3B%0A%20%20%
  20%20%20%20min_index%20%3D%20i%3B%0A%20%20%20%20%7D%0A%20%
  20%7D%0A%0A%20%20return%20min_index%3B%0A%7D%0A%0Avoid%
  20WrongSwap%28int%20left,%20int%20right%29%20%7B%0A%20%
  20int%20temp%20%3D%20left%3B%0A%20%20left%20%3D%20right%3B
  %0A%20%20right%20%3D%20temp%3B%0A%7D%0A%0Avoid%20Swap%
  28int%20*left,%20int%20*right%29%20%7B%0A%20%20int%20temp%
  20%3D%20*left%3B%0A%20%20*left%20%3D%20*right%3B%0A%20%20*
  right%20%3D%20temp%3B%0A%7D%0A%0Avoid%20Print%28const%
  20int%20arr%5B%5D,%20int%20len%29%20%7B%0A%20%20printf%28%
  22%5Cn%22%29%3B%0A%20%20for%20%28int%20i%20%3D%200%3B%20i%
  20%3C%20len%3B%20i%2B%2B%29%20%7B%0A%20%20%20%20printf%28%
  22%25d%20%22,%20arr%5Bi%5D%29%3B%0A%20%20%7D%0A%20%
  20printf%28%22%5Cn%22%29%3B%0A%7D&cumulative=true&curInstr
  =0&heapPrimitives=nevernest&mode=display&origin=opt-
  frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D&
  textReferences=false
4 // Visualization of malloc:
```

```c

#include <stdio.h>
#include <stdlib.h>

#define LEN 5

void SelectionSort(int arr[], int len);
void WrongSwap(int left, int right);
void Swap(int *left, int *right);
int GetMinIndex(const int arr[], int begin, int end);
void Print(const int arr[], int len);

int main(void) {
  int len = 0;
  scanf("%d", &len);
  // (void *)
  // size_t size: unsigned long/long long
  // malloc vs. calloc:
  // https://stackoverflow.com/questions/1538420/difference-between-malloc-and-calloc
  int *numbers = malloc(len * sizeof(*numbers));

  // NULL: null pointer ((void *) 0) in the C standards
  if (numbers == NULL) {
    return 0;
  }

  for (int i = 0; i < len; ++i) {
    scanf("%d", &numbers[i]);
  }

  Print(numbers, len);
  // (): function-call operator
  SelectionSort(numbers, len);
  // SelectionSort(&numbers[0], LEN);
  Print(numbers, len);

  free(numbers);
  // free(numbers);
  // numbers[3] = 5;
}

// arr: the (copy of the) address of the first element of the `numbers` array
```

```c
// int arr[] <=> int *arr
void SelectionSort(int *arr, int len) {
  for (int i = 0; i < len; i++) {
    int min_index = GetMinIndex(arr, i, len);
    // &arr[i] <=> &(*(arr + i)) <=> arr + i
    Swap(arr + i, arr + min_index);
  }
}

int GetMinIndex(const int *arr, int begin, int end) {
  int min = arr[begin];
  int min_index = begin;

  for (int i = begin + 1; i < end; ++i) {
    // arr[i] <=> *(arr + i) <=> *(i + arr) <=> i[arr]
    // arr + i, arr - i, p - q
    if (arr[i] < min) {
      min = arr[i];
      min_index = i;
    }
  }

  return min_index;
}

void WrongSwap(int left, int right) {
  int temp = left;
  left = right;
  right = temp;
}

void Swap(int *left, int *right) {
  int temp = *left;
  *left = *right;
  *right = temp;
}

void Print(const int arr[], int len) {
  printf("\n");
  for (int i = 0; i < len; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");
}
```