|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | 1. Make a Calendar (Horizontally as well as Vertically) | | #include<iostream>  #include<string>  using namespace std;  bool is\_leap(int year){  return (year % 400 == 0) || (year % 100 != 0 && year % 4 == 0);  }  int print\_month\_horizontally(int n\_days, int start\_day){  cout<<"Mon\tTue\tWed\tThu\tFri\tSat\tSun\n";  for(int j=0; j<start\_day; j++){  cout<<"\t";  }  for (int i=1; i<=n\_days; i++){  cout<<i<<"\t";  if ((i - 7 + start\_day) % 7 == 0){  cout<<"\n";  }  }  int last\_day = (start\_day + n\_days) % 7;  return last\_day;  }  int print\_month\_vertically(int n\_days, int start\_day){  string week[7] = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};  for (int i=0; i<7; i++){  cout<<week[i]<<"\t";  if (i < start\_day) {  cout<<"\t";  int a = 7-start\_day +1 + i;  while (a <= n\_days){  cout<<a<<"\t";  a = 7+a;  }  }  else  {  int b = i - start\_day + 1;  while (b <= n\_days){  cout<<b<<"\t";  b = 7+b;  }  }  cout<<"\n";    }  int last\_day = (start\_day + n\_days) % 7;  return last\_day;  }  int first\_day(int data, int year){  if (year >= 2020){ // since data is of 2020 (reference year) we use 2020  int x = year - 2020;  int counter=0;  for (int i=2020; i<year; i++){  if (is\_leap(i)){  counter++;  }  }  int norm = x - counter;  return (data + norm + 2\*counter) % 7;  }else if(year == 2020){  return data;  } else {  int x = 2020 - year;  int counter = 0;  for (int i=year; i<2020; i++){  if (is\_leap(i)){  counter++;  }  }  int norm = x - counter;  return (7 - ((data - norm - 2\*counter) % 7)) % 7;  }  }    int main(){  int data = 2; // 2 means wednesday which was Jan 1, 2020  int year, feb\_day=28;  cout<<"Enter year: ";  cin>>year;  if (is\_leap(year)){  feb\_day=29;  }  int first = first\_day(data, year);  cout<<"\nHORIZONTAL CALENDAR\n--------------------------";  cout<<"\n\nJanuary\n\n";  first = print\_month\_horizontally(31, first); //this function gives the day after last date in that month  cout<<"\n\nFebruary\n\n";  first = print\_month\_horizontally(feb\_day, first);  cout<<"\n\nMarch\n\n";  first = print\_month\_horizontally(31, first);  cout<<"\n\nApril\n\n";  first = print\_month\_horizontally(30, first);  cout<<"\n\nMay\n\n";  first = print\_month\_horizontally(31, first);  cout<<"\n\nJune\n\n";  first = print\_month\_horizontally(30, first);  cout<<"\n\nJuly\n\n";  first = print\_month\_horizontally(31, first);  cout<<"\n\nAugust\n\n";  first = print\_month\_horizontally(31, first);  cout<<"\n\nSeptember\n\n";  first = print\_month\_horizontally(30, first);  cout<<"\n\nOctober\n\n";  first = print\_month\_horizontally(31, first);  cout<<"\n\nNovember\n\n";  first = print\_month\_horizontally(30, first);  cout<<"\n\nDecember\n\n";  first = print\_month\_horizontally(31, first);  cout<<"\nVERTICAL CALENDAR\n--------------------------";  first = first\_day(data, year);  cout<<"\n\nJanuary\n\n";  first = print\_month\_vertically(31, first); //this function gives the day after last date in that month  cout<<"\n\nFebruary\n\n";  first = print\_month\_vertically(feb\_day, first);  cout<<"\n\nMarch\n\n";  first = print\_month\_vertically(31, first);  cout<<"\n\nApril\n\n";  first = print\_month\_vertically(30, first);  cout<<"\n\nMay\n\n";  first = print\_month\_vertically(31, first);  cout<<"\n\nJune\n\n";  first = print\_month\_vertically(30, first);  cout<<"\n\nJuly\n\n";  first = print\_month\_vertically(31, first);  cout<<"\n\nAugust\n\n";  first = print\_month\_vertically(31, first);  cout<<"\n\nSeptember\n\n";  first = print\_month\_vertically(30, first);  cout<<"\n\nOctober\n\n";  first = print\_month\_vertically(31, first);  cout<<"\n\nNovember\n\n";  first = print\_month\_vertically(30, first);  cout<<"\n\nDecember\n\n";  first = print\_month\_vertically(31, first);    return 0;  } | |  | |
| |  | | --- | | 1. Given array of integers, arrange the array such that even integers are on one side and odd ones in the other. Don’t use another array. | | Solution 1:  #include<iostream>  using namespace std;  void swape(int \*x, int \*y){  int temp;  temp = \*x;  \*x = \*y;  \*y = temp;  }  int main(){  int a[30] = {21, 44, 65, 35, 65, 86, 254, 75, 70, 57, 35, 44, 83, 92, 84, 54, 20, 65, 73, 24, 46, 923, 934, 43, 2, 59 ,27, 40, 02, 46};  int n = 30;  int swaps = 1;  while(swaps != 0){  swaps = 0;  for (int l=0, h=1; l<n-1, h<n; l++, h++){  if (a[l] % 2 != 0 && a[h] % 2 == 0){  swape(&a[l], &a[h]);  swaps++;  }  }  }  for (int i=0; i<n; i++){cout<<a[i]<<" ";}  cout<<"\n";  return 0;  } | |  | | Solution 2:  #include<iostream>  using namespace std;  void swape(int \*x, int \*y){  int temp;  temp = \*x;  \*x = \*y;  \*y = temp;  }  int main(){  int a[30] = {21, 44, 65, 35, 65, 86, 254, 75, 70, 57, 35, 44, 83, 92, 84, 54, 20, 65, 73, 24, 46, 923, 934, 43, 2, 59 ,27, 40, 02, 46};  int n = 30;  // Given below is another solution which use way less swaps  bool over = false;  int swapes = 0;  for (int i=0; i<n; i++){  if (a[i] % 2 != 0){  for (int j=n-1; j >= i; j--){  if (a[j] % 2 == 0){  swape(&a[i], &a[j]);  swapes++;  break;  } else if (j==i) {  over = true;  }  }  }  if (over){  break;  }  }  for (int i=0; i<n; i++){cout<<a[i]<<" ";}  cout<<"\n"<<swapes;  return 0;  } | |  | |
| |  | | --- | | 1. Write the swap function in multiple ways. | | #include<iostream>  using namespace std;  void swap1(int x, int y){ //call-by-value  int temp = x;  x = y;  y = temp;  }  void swap2(int &x, int &y){ // call-by-reference Using Alias x and y  int temp = x;  x = y;  y = temp;  }  void swap3(int \*x, int \*y) { // call-by-pointer  int temp = \*x;  \*x = \*y;  \*y = temp;  }  void swap4(int &x, int \*y) {  int temp = x;  x = \*y;  \*y = temp;  }  void swap5(int \*x, int &y) {  int temp = \*x;  \*x = y;  y = temp;  }  // void swap6(int \*x, int \*y) {  // int \*temp;  // \*temp = \*x; // Assigning value to a pointer that wasn't intialised  // // is not encouraged however newer compilers will work  // // in my machine it crashed  // \*x = \*y;  // \*y = \*temp;  // }  int main(){  int a=3, b=6;  cout<<"a = "<<a<<", b = "<<b<<endl;  swap1(a, b); // Since it is call by value It won't swap the original a and b.  cout<<"After Swap1 - Call By Value"<<endl;  cout<<"a = "<<a<<", b = "<<b<<endl;  swap2(a, b);  cout<<"After Swap2 - Call by Reference"<<endl;  cout<<"a = "<<a<<", b = "<<b<<endl;  swap3(&a, &b);  cout<<"After Swap3 - Call by Pointer"<<endl;  cout<<"a = "<<a<<", b = "<<b<<endl;  swap4(a, &b);  cout<<"After Swap4 - Call by reference and pointer"<<endl;  cout<<"a = "<<a<<", b = "<<b<<endl;  swap5(&a, b);  cout<<"After Swap5 - Call by pointer and reference"<<endl;  cout<<"a = "<<a<<", b = "<<b<<endl;  return 0;  } | |  | |