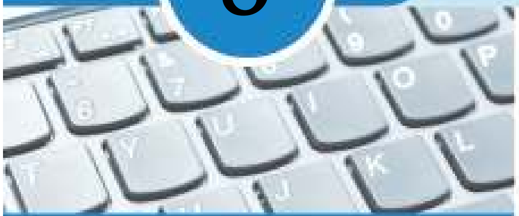
8



പ്രധാന ആശയങ്ങൾ

- അറെയും ആവശ്യകതയും
  - അറെ പ്രഖ്യാപനം
  - അറെയുടെ മെമ്മറി നീക്കി വയ്ക്കൽ
  - അറെയും പ്രാരംഭ വില നൽകലും
  - അറെയിലെ അംഗങ്ങളെ ഉപയോഗിക്കൽ
- അറെയുടെ പ്രവർത്തനങ്ങൾ
  - കടന്നുപോകൽ
  - ക്രമപ്പെടുത്തൽ
    - സെലക്ഷൻ സോർട്ട്
    - ബബിൾ സോർട്ട്
  - തിരയൽ
    - രേഖീയ തിരയൽ
    - ബൈനറി തിരയൽ
- ദ്വിമാന അറെകൾ
  - ദ്വിമാന അറെ പ്രഖ്യാപനം
  - മെട്രിക്സായി ദ്വിമാന അറെകൾ
- ബഹുമുഖ അറെകൾ

Y6X6M2

അറെകൾ

പ്രോഗ്രാമുകളിൽ ഡാറ്റ സംഭരിക്കുന്നതിനായി നാം വേരിയബലുകൾ ഉപയോഗിക്കുന്നു. എന്നാൽ ഡാറ്റയുടെ എണ്ണം കൂടുതലാണെങ്കിൽ കൂടുതൽ വേരിയബലുകൾ ഉപയോഗിക്കേണ്ടതായി വരും. ഈ സാഹചര്യത്തിൽ ഡാറ്റ ഉപയോഗിക്കുന്ന രീതി വളരെ ബുദ്ധിമുട്ടുള്ളതായി അനുഭവപ്പെടും. ഇതു മറികടക്കാൻ ഈ അദ്ധ്യായത്തിൽ അറെ (Array) എന്ന പേരിലുള്ള C++ ൽ നിന്നും ഉരുത്തിരിഞ്ഞ ഡാറ്റ ഇനം പരിചയപ്പെടുത്തുന്നു. അറെ എന്നത് കേവലമൊരു ഡാറ്റ ഇനത്തിന്റെ നാമം മാത്രമല്ല, മറിച്ച് ഇത് വളരെ കൂടുതൽ ഡാറ്റ എളുപ്പത്തിൽ കൈകാര്യം ചെയ്യുന്നതിന് വേണ്ടി അടിസ്ഥാനപരമായ ഡാറ്റ ഇനങ്ങളിൽ നിന്നും നിർമിച്ചെടുത്ത മറ്റൊരു തരം ഡാറ്റ ഇനമാണ്. അറെയുടെ പ്രഖ്യാപനം പ്രാഥമിക വിലയിരുത്തൽ (Initialization), കടന്നുപോകൽ (Traversal), ക്രമപ്പെടുത്തൽ (Sorting), തിരയൽ (Searching) പോലുള്ള പ്രവർത്തനങ്ങളെപ്പറ്റി നമുക്ക് ചർച്ച ചെയ്യാം.

8.1 അറെയും അവയുടെ ആവശ്യകതയും (Array and its need)

അറെ എന്നാൽ തുടർച്ചയായ മെമ്മറി സ്ഥാനങ്ങളിൽ ശേഖരിച്ചു വച്ചിട്ടുള്ള ഒരേ ഇനത്തിലുള്ള ഡാറ്റകളുടെ സമൂഹമാണ്. ഒരു പേരിൽ ഒരേ ഇനത്തിലുള്ള ഒരു കൂട്ടം വിലകൾ ശേഖരിക്കുന്നതിനായി അറെകൾ ഉപയോഗിക്കുന്നു. ഒരു അറെയിലെ ഓരോ അംഗങ്ങളേയും അതിന്റേതായ സൂചിക വ്യക്തമാക്കിക്കൊണ്ട് ഉപയോഗിക്കുവാൻ സാധിക്കും.

എന്തുകൊണ്ടാണ് പ്രോഗ്രാമുകളിൽ അറെ ആവശ്യമായിവരുന്നത്. ഒരു ഉദാഹരണത്തിന്റെ സഹായത്തോടെ ഇത് നമുക്ക് പരിശോധിക്കാം. ഒരു ക്ലാസിലെ 20 വിദ്യാർത്ഥികളുടെ മാർക്കുകളുടെ ശരാശരിയെ കണ്ടെത്തണം എന്ന് കരുതുക. ഈ സാഹചര്യത്തിൽ സാധാരണ വേരിയബിളുകൾ ഉപയോഗിച്ചാൽ 20 വിദ്യാർത്ഥികളുടെ മാർക്കുകൾ ശേഖരിക്കുവാൻ 20 വേരിയബിളുകൾ ആവശ്യമായി വരും.

```
int  a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t;
float avg;
cin>>a>>b>>c>>d>>e>>f>>g>>h>>i>>j>>k>>l>>m>>n>>o>>p>>q>>r>>s>>t;
avg  =  (a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t)/20.0;
```

ഒരു പരിധിവരെ മുകളിൽ കൊടുത്തിരിക്കുന്ന കോഡ് ഉപയോഗിച്ച് 20 കുട്ടികളുടെ മാർക്കുകളുടെ ശരാശരി കണ്ടുപിടിക്കുവാൻ കഴിയും. എന്നാൽ 1000 കുട്ടികളുടെ ശരാശരി മാർക്ക് കണ്ടുപിടിക്കേണ്ട ഒരു സാഹചര്യം ഉണ്ടായാൽ ഈ രീതിയിലുള്ള പ്രവർത്തനം സാധ്യമല്ല. അതായത് ഒരു പ്രോഗ്രാമിൽ 1000 വേരിയബിളുകൾ ഉപയോഗിക്കുന്നതും അവ ഉപയോഗിച്ച് പ്രോഗ്രാം ചെയ്യുന്നതും എളുപ്പമുള്ള കാര്യമല്ല, മാത്രമല്ല ഇങ്ങനെ നിർമിക്കുന്ന പ്രോഗ്രാം വളരെ സങ്കീർണവും മനസ്സിലാക്കുന്നതിന് ബുദ്ധിമുട്ടുള്ളതും ആയിരിക്കും. ഇത്തരം സാഹചര്യങ്ങളിൽ അറേ എന്ന ആശയം നമുക്ക് ഉപകരിക്കും. അറേയിലെ ഓരോ അംഗങ്ങൾക്കും മെമ്മറി സ്ഥാനങ്ങൾ അനുവദിക്കേണ്ടതുണ്ട്. മെമ്മറി നീക്കിവെയ്ക്കുന്നതിന് പ്രഖ്യാപന പ്രസ്താവനകൾ ആവശ്യമാണെന്നും നമുക്കറിയാം. എങ്ങനെയാണ് അറേകൾ പ്രഖ്യാപനം നടത്തി അവ ഉപയോഗിക്കുന്നത് എന്ന് നമുക്ക് നോക്കാം.

8.1.1 അറേകളുടെ പ്രഖ്യാപനം (Array Declaration)

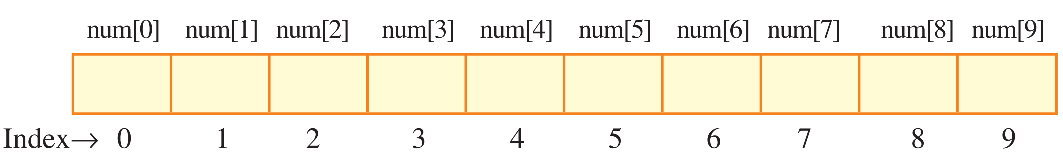
സാധാരണ വേരിയബിളിനെ പോലെ അറേ ഉപയോഗിക്കുന്നതിന് മുമ്പായി പ്രഖ്യാപനം നടത്തേണ്ടതുണ്ട്. C++ൽ അറേ പ്രഖ്യാപനം ചെയ്യുന്നതിനുള്ള വാക്യഘടന താഴെ പറഞ്ഞിരിക്കുന്നു.

```
datatype  array_name[size];
```

വാക്യഘടനയിൽ datatype എന്നത് അറേയിലെ അംഗങ്ങളുടെ ഡേറ്റയുടെ ഇനമാണ് സൂചിപ്പിക്കുന്നത്. array_name എന്നത് അറേയുടെ പേരും size എന്നത് അറേയിലെ ആകെ അംഗങ്ങളുടെ എണ്ണം വ്യക്തമാക്കുന്ന ഒരു പോസിറ്റീവ് സംഖ്യയും ആകുന്നു. താഴെ പറയുന്നത് ഒരു അറേ നിർമാണത്തിന്റെ ഉദാഹരണമാണ്.

```
int  num[10];
```

മുകളിലുള്ള പ്രസ്താവന num എന്ന് വിളിക്കുന്ന 10 പൂർണ്ണസംഖ്യകൾ സൂക്ഷിക്കാവുന്ന ഒരു അറേയെ നിർമിക്കുന്നു. ചിത്രം 8.1 കാണിച്ചിരിക്കുന്നതു പോലെ അറേയിലെ അംഗങ്ങൾ മെമ്മറിയിൽ തുടർച്ചയായി സൂക്ഷിക്കുന്നു.



ചിത്രം 8.1 ഒരു അറേയിലെ അംഗങ്ങളുടെ ക്രമീകരണം

അറേയിലെ അംഗങ്ങൾ ക്രമാനുഗതമായി സൂക്ഷിക്കുന്നതുകൊണ്ട്, ഏത് അംഗത്തേയും അറേയുടെ പേരും അംഗത്തിന്റെ സ്ഥാനവും നൽകി ഉപയോഗിക്കുവാൻ കഴിയും. ഓരോ അംഗത്തെയും സൂചിപ്പിക്കുന്ന സ്ഥാനത്തിന് സൂചിക (index or subscript) എന്നു പറയുന്നു.

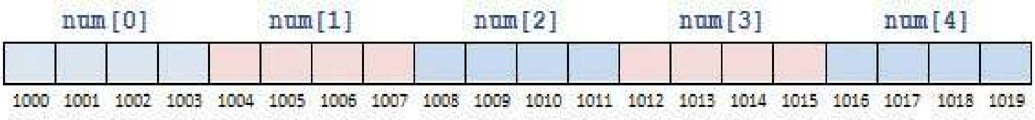
C++ൽ അറെയുടെ സൂചിക പൂജ്യത്തിൽ ആരംഭിക്കുന്നു. `int num[10]` എന്ന് ഒരു അറെ നിർമ്മിച്ചാൽ അതിൽ സാധ്യമായ സൂചിക വിലകൾ 0 മുതൽ 9 വരെയാകും. ഈ അറെയിലെ ഒന്നാമത്തെ അംഗം `num [0]` ഉം അവസാനത്തെ അംഗം `num [9]` ഉം ആകുന്നു. `num [0]` എന്നത് 'നം ഓഫ് സീറോ' എന്ന് വായിക്കുന്നു. ആയിരം വിദ്യാർത്ഥികളുടെ മാർക്കുകൾ സംഭരിക്കുന്ന പ്രശ്നം താഴെപ്പറയുന്ന പ്രസ്താവന ഉപയോഗിച്ച് പരിഹരിക്കാനാകും.

```
int score[1000];
```

`score` എന്നു പേരുള്ള അറെയിൽ 1000 വിദ്യാർഥികളുടെ മാർക്കുകൾ സംഭരിക്കാം. ആദ്യ വിദ്യാർത്ഥിയുടെ മാർക്ക് `score[0]` ലും അവസാനത്തെ വിദ്യാർത്ഥിയുടെ മാർക്ക് `score[999]` ലും സംഭരിക്കും.

8.1.2 അറെയുടെ മെമ്മറി നീക്കിവെയ്ക്കൽ (Memory Allocation for Arrays)

ഒരു അറെയിൽ അംഗങ്ങളെ സംഭരിക്കുന്നതിന് ആവശ്യമായ മെമ്മറിയുടെ അളവ് അതിന്റെ ഇനവും അംഗങ്ങളുടെ എണ്ണവുമായി ബന്ധപ്പെട്ടിരിക്കുന്നു. ചിത്രം 8.2ൽ `num` എന്ന ഒരു അറെയുടെ മെമ്മറി നീക്കിവെയ്ക്കൽ കാണിച്ചിരിക്കുന്നു, ഇതിൽ ആദ്യ അംഗത്തിന്റെ വിലാസമായി 1000 എന്ന് കാണിച്ചിരിക്കുന്നു. `num` ഒരു പൂർണ്ണസംഖ്യകളുടെ അറെ ആയ തിനാൽ, ഓരോ അംഗത്തിന്റെയും വ്യാപ്തി 4 ബൈറ്റുകൾ ആണ് (16 ബിറ്റ് പ്രതിനി ധീകരിക്കുന്ന ഒരു സിസ്റ്റത്തിൽ). താഴെക്കൊടുത്തിരിക്കുന്ന ചിത്രം 8.2ൽ `num[0]`ന്റെ വിലാസം `1000`, `num[1]`ന്റെ വിലാസം `1004`, അവസാന അംഗമായ `num[4]` വിലാസം വിലാസം `1016` എന്നിങ്ങനെ ആയിരിക്കും.



ചിത്രം 8.2 ഒരു പൂർണ്ണ സംഖ്യ അറെയുടെ മെമ്മറി അലോക്കേഷൻ

ഒരു ഏകമാന അറെക്ക് (single dimensional array) ആവശ്യമായ മെമ്മറിയുടെ അളവ് താഴെ പറയുന്ന സൂത്രവാക്യം ഉപയോഗിച്ച് കണ്ടുപിടിക്കാം.

ആകെ ബൈറ്റുകൾ = size_of (അറെയുടെ ഇനം) × അറെയിലെ അംഗങ്ങളുടെ എണ്ണം

ഉദാഹരണത്തിന്, `int num [10];` `num` അറെക്കായി നീക്കിവെച്ചിട്ടുള്ള ആകെ ബൈറ്റുകൾ $4 \times 10 = 40$ ബൈറ്റുകൾ ആയിരിക്കും.

8.1.3 അറെയുടെ പ്രാരംഭ വില നൽകൽ (Array Initialization)

സാധാരണ വേരിയബിൾ പോലെ തന്നെ അറെയുടെ പ്രഖ്യാപന പ്രസ്താവനകളോ ടൊപ്പം അവയുടെ പ്രാരംഭ വിലകൾ നൽകുവാൻ കഴിയും. താഴെപ്പറയുന്ന ഉദാഹരണ ങ്ങളിൽ കാണിച്ചിരിക്കുന്നതുപോലെ അറെയിലെ അംഗങ്ങളെ ബ്രാക്കറ്റിനുള്ളിൽ എഴു തണം.

```
int score[5] = {98, 87, 92, 79, 85};
char code[6] = {'s', 'a', 'm', 'p', 'l', 'e'};
float wgpa[7] = {9.60, 6.43, 8.50, 8.65, 5.89, 7.56, 8.22};
```

അറേയിലെ അംഗങ്ങളെ അവ എഴുതപ്പെട്ട ക്രമത്തിൽ സൂക്ഷിക്കുന്നു. ഒന്നാമത്തെ അംഗം സൂചിക 0ലും, രണ്ടാമത്തെ അംഗം സൂചിക 1ലും പ്രാരംഭ വിലകളായി സൂക്ഷിക്കുന്നു. ആദ്യത്തെ ഉദാഹരണത്തിൽ, `score[0]` ലേക്ക് 98, `score[1]` ലേക്ക് 87, `score[2]` ലേക്ക് 92, `score[3]` ലേക്ക് 79, `score[4]` ലേക്ക് 85 ഉം പ്രാരംഭ വിലകളായി സൂക്ഷിക്കുന്നു. ഒരു അറെയ്ക്ക് അനുവദിക്കപ്പെട്ട അംഗങ്ങളുടെ എണ്ണത്തെക്കാൾ പ്രാരംഭ മൂല്യങ്ങളുടെ എണ്ണം കുറവാണെങ്കിൽ, ആദ്യ സ്ഥാനങ്ങളിൽ അംഗങ്ങൾ സംഭരിക്കും, ശേഷിക്കുന്ന സ്ഥാനങ്ങൾ സംഖ്യാ ഡാറ്റകളുടെ കാര്യത്തിൽ പൂജ്യവും അക്ഷരഡാറ്റകളുടെ കാര്യത്തിൽ ' ' (സ്പെയിസും) സംഭരിക്കും. ഒരു അറേയിലെ അംഗങ്ങളുടെ പ്രാരംഭ വിലകൾ നൽകുമ്പോൾ അംഗങ്ങളുടെ എണ്ണം ഒഴിവാക്കാവുന്നതാണ്. ഉദാഹരണത്തിന്, താഴെ പറയുന്ന പ്രാരംഭ വില നൽകൽ പ്രസ്താവന അഞ്ച് അംഗങ്ങളുള്ള ഒരു അറെ നിർമ്മിക്കുന്നു.

```
int num[] = {16, 12, 10, 14, 11};
```

8.1.4 അറെയിലെ അംഗങ്ങളെ ഉപയോഗിക്കൽ (Accessing elements of arrays)

ഒരു അറെയിലെ അംഗങ്ങളെ പ്രോഗ്രാമിൽ എവിടെയും ഉപയോഗിക്കാം. ഒരു സമയം ഒരു അംഗത്തിനെ മാത്രമേ ഉപയോഗിക്കാൻ കഴിയു. ഓരോ അംഗത്തേയും അറെയുടെ പേരും അവയുടെ സൂചികയും നല്കി ഉപയോഗിക്കുന്നു. `score` എന്ന അറെയിലെ അംഗങ്ങളെ ഉപയോഗിക്കുന്ന ചില ഉദാഹരണങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നു.

```
score[0] = 95;
score[1] = score[0] - 11;
cin >> score[2];
score[3] = 79;
cout << score[2];
sum = score[0] + score[1] + score[2] + score[3] + score[4];
```

ബ്രാക്കറ്റിനുള്ളിലെ സൂചിക ഒരു വേരിയബിളോ, ഒരു പൂർണ്ണസംഖ്യയോ, പൂർണ്ണസംഖ്യ നിർദ്ധാരണം ചെയ്യുന്ന ഒരു പ്രസ്താവനയോ ആകാം. ഓരോ സന്ദർഭത്തിലും പ്രസ്താവനയുടെ മൂല്യം അറെയുടെ സൂചികയുടെ സാധുവായ പരിധിക്കുള്ളിൽ ആയിരിക്കണം. ഈ രീതിയിൽ വേരിയബിളോ പ്രസ്താവനയോ ഉപയോഗിക്കുന്നതിന്റെ ഗുണം, അറെയിലുള്ള അംഗങ്ങളെ ഉപയോഗിക്കുന്നതിന് വേണ്ടി ലൂപ്പിന്റെ നിയന്ത്രണ വേരിയബിള് ഉപയോഗിക്കാം എന്നുള്ളതാണ്. ഇത് പ്രസ്താവനകളെ താഴെപ്പറയുന്ന രീതിയിൽ അനുചിതമായി ഉപയോഗിക്കുന്നതിൽ നിന്നും നമ്മെ പിന്തിരിപ്പിക്കുന്നു.

```
sum = score[0] + score[1] + score[2] + score[3] + score[4];
```

മുകളിലുള്ള പ്രസ്താവനയിലെ സൂചികയുടെ മൂല്യങ്ങൾക്കു പകരം ലൂപ്പിന്റെ നിയന്ത്രണ വേരിയബിൾ ഉപയോഗിച്ചുകൊണ്ട് അറെയിലെ അംഗങ്ങളെ ഉപയോഗിക്കാം. താഴെപ്പറയുന്ന പ്രസ്താവനകൾ ഈ ആശയം വിശദമാക്കുന്നു.

```
sum = 0;
for (i=0; i<5; i++)
sum = sum + score[i];
```

താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ ഒരു ഇൻപുട്ട് പ്രസ്താവന ഉപയോഗിച്ചുകൊണ്ടും അറെയിലെ അംഗത്തിന് മൂല്യം നൽകാം.

```
for(int i=0; i<5; i++)
    cin>>score[i];
```

ഈ ലൂപ്പ് പ്രവർത്തിച്ചു കഴിയുമ്പോൾ ആദ്യം സ്വീകരിക്കുന്ന വില അറെയുടെ ഒന്നാമത്തെ അംഗമായ `score [0]` ലും, രണ്ടാമത്തെ വില `score [1]` ലും, അവസാന വില `score[4]` ലും സൂക്ഷിക്കുന്നു.

പ്രോഗ്രാം 8.1 ഒരു അറെയിൽ എങ്ങനെ അഞ്ച് വിലകൾ സ്വീകരിക്കാമെന്നും അവയെ വിപരീത ക്രമത്തിൽ പ്രദർശിപ്പിക്കാമെന്നും കാണിക്കുന്നു. ഈ പ്രോഗ്രാമിൽ ഉൾപ്പെടുത്തിയിട്ടുള്ള രണ്ട് ലൂപ്പുകളിൽ ആദ്യത്തേത് അറെയുടെ അംഗങ്ങളുടെ വിലകൾ സ്വീകരിക്കുന്നു. അഞ്ച് വിലകൾ സ്വീകരിച്ച് കഴിഞ്ഞാൽ രണ്ടാമത്തെ ലൂപ്പ് സംഭരിച്ച വിലകളെ അവസാനം മുതൽ ആദ്യം വരെ പ്രദർശിപ്പിക്കുന്നു.

പ്രോഗ്രാം 8.1: 5 കുട്ടികളുടെ സ്കോറുകൾ ഇൻപുട്ട് ചെയ്ത്, അവയെ നേർവിപരീത ക്രമത്തിൽ പ്രദർശിപ്പിക്കുക.

```
#include <iostream>
using namespace std;
int main()
{
  int i, score[5];
  for(i=0; i<5; i++)   // Reads the scores
  {
    cout<<"Enter a score: ";
    cin>>score[i];
  }
  for(i=4; i>=0; i--)   // Prints the scores
    cout<<"score[" << i << "] is " << score[i]<<endl;
  return 0;
}
```

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
Enter a score: 55
Enter a score: 80
Enter a score: 78
Enter a score: 75
Enter a score: 92
score[4] is 92
score[3] is 75
```

```
score[2] is 78
score[1] is 80
score[0] is 55
```

നമുക്ക് ചെയ്യാം

1. താഴെ പറയുന്നവ സംഭരിക്കുന്നതിനുള്ള അറെ പ്രഖ്യാപന പ്രസ്താവനകൾ എഴുതുക
   i. 100 വിദ്യാർത്ഥികളുടെ മാർക്ക്
   ii. ഇംഗ്ലീഷ് അക്ഷരമാല
   iii. 10 വർഷങ്ങളുടെ പട്ടിക
   iv. 30 ദശാംശ സംഖ്യകളുടെ പട്ടിക
2. താഴെ പറയുന്ന അറെയിൽ പ്രാരംഭ വിലകൾ നല്കുന്നതിനുള്ള പ്രസ്താവനകൾ എഴുതുക
   i. 10 സ്കോറുകളുടെ പട്ടിക 89, 75, 82, 93, 78, 95, 81, 88, 77, 82
   ii. അഞ്ച് അളവുകളുടെ പട്ടിക: 10.62, 13.98, 18.45, 12.68, 14.76 എന്നിവ
   iii. 100 പലിശ നിരക്കുകളുടെ പട്ടിക, ആദ്യ ആറ് പലിശ നിരക്കുകൾ 6.29, 6.95, 7.25, 7.35, 7.40, 7.42.
   iv. മൂല്യം 0 ഉപയോഗിച്ച് 10 മാർക്കിനുള്ള ഒരു അറെ.
   v. VIBGYOR അക്ഷരങ്ങളുള്ള ഒരു അറെ.
   vi. ഓരോ മാസത്തിലുമുള്ള ദിവസങ്ങളുള്ള ഒരു അറെ.
3. `int ar[50];` എന്ന അറെയിലേക്ക് വിലകൾ ഇൻപുട്ട് ചെയ്യുന്നതിനുള്ള C++ കോഡ് ശകലങ്ങൾ എഴുതുക.
4. `float val [100]; val` അറെയുടെ ഇരട്ട സ്ഥാനങ്ങളിലുള്ള അംഗങ്ങൾ പ്രദർശിപ്പിക്കുന്നതിന് C++ കോഡ് ശകലം എഴുതുക:

8.2 അറെയുടെ പ്രവർത്തനങ്ങൾ (Array Operations)

കടന്നുപോകൽ (Traversal), ക്രമപ്പെടുത്തൽ (Sorting), തിരയൽ (Searching), ഇടയിൽ ചേർക്കൽ (Insertion), നീക്കം ചെയ്യൽ (Deletion), ലയിപ്പിക്കൽ (Merging) തുടങ്ങിയവ അറെയുടെ പ്രവർത്തനങ്ങളിൽ ഉൾപ്പെടുന്നു. ഈ പ്രവർത്തനങ്ങൾ നടത്താൻ വ്യത്യസ്തയുക്തികൾ പ്രയോഗിക്കുന്നു. അവയിൽ ചിലത് നമുക്ക് ചർച്ചചെയ്യാം.

8.2.1 കടന്നുപോകൽ (Traversal)

കുറഞ്ഞത് ഒരിക്കലെങ്കിലും അറെയിലെ ഓരോ അംഗത്തേയും ഉപയോഗിക്കുക എന്നതാണ് കടന്നുപോകൽ എന്നതുകൊണ്ട് ഉദ്ദേശിക്കുന്നത്. ഇടയിൽ ചേർക്കൽ, നീക്കം ചെയ്യൽ തുടങ്ങിയ പ്രവർത്തനങ്ങളുടെ കൃത്യത പരിശോധിക്കുവാൻ കടന്നുപോകൽ പ്രവർത്തനം നമുക്ക് ഉപയോഗിക്കാം. അറെയിലെ എല്ലാ അംഗങ്ങളേയും പ്രദർശിപ്പിക്കുന്നത് കടന്നുപോകലിന് ഒരു ഉദാഹരണമാണ്. ഏതെങ്കിലുമൊരു പ്രവർത്തനം അറെയിലെ എല്ലാ അംഗങ്ങളിലും നടക്കുന്നു എങ്കിൽ അതിനെ കടന്നുപോകൽ എന്നു പറയുന്നു .

ഒരു പ്രോഗ്രാമിൽ എങ്ങനെയാണ് കടന്നുപോകൽ നടത്തുന്നത് എന്നത് താഴെ കൊടുത്തിരിക്കുന്നു

പ്രോഗ്രാം 8.2: അറെയിലെ കടന്നുപോകൽ

```
#include <iostream>
using namespace std;
int main()
{
int a[10], i;
cout<<"Enter the elements of the array :";
for(i=0; i<10; i++)
      cin >> a[i];
for(i=0; i<10; i++)
      a[i] = a[i] + 1;
cout<<"\nEntered elements of the array are...\n";
for(i=0; i<10; i++)
     cout<< a[i]<< "\t";
return 0;
}
```

(ഓരോ for ലൂപ്പിനും ചേർത്ത അടയാളം: കടന്നുപോകൽ)

8.2.2 ക്രമപ്പെടുത്തൽ (Sorting)

ചില യുക്തിപരമായ ക്രമത്തിൽ അറെയിലെ അംഗങ്ങൾ ക്രമീകരിക്കുന്ന പ്രവർത്തനമാണ് ക്രമപ്പെടുത്തൽ. വാക്കുകളുടെ കാര്യത്തിൽ നിഘണ്ടു ക്രമവും സംഖ്യകളുടെ കാര്യത്തിൽ അവയുടെ മൂല്യങ്ങളുടെ ആരോഹണ ക്രമമോ അവരോഹണ ക്രമമോ ആകാം യുക്തി പരമായ ക്രമം. ഈ പ്രവർത്തനം ഫലപ്രദമായി ചെയ്യാൻ പല അൽഗോരിതങ്ങളുമുണ്ട് ഇവയിൽ സെലക്ഷൻ സോർട്ട്, ബബിൾ സോർട്ട് എന്നീ അൽഗോരിതങ്ങൾ നമുക്ക് ഇവിടെ ചർച്ചചെയ്യാം.

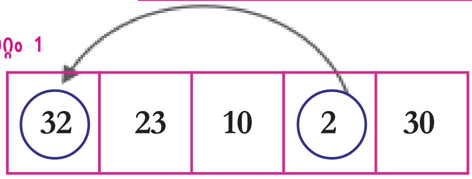
a. സെലക്ഷൻ സോർട്ട് (Selection sort)

ഏറ്റവും ലളിതമായ ക്രമപ്പെടുത്തൽ രീതികളിലൊന്നാണ് സെലക്ഷൻ സോർട്ട്. ആരോഹണക്രമത്തിൽ ഒരു അറെ ക്രമീകരിക്കുന്നതിനായി അറെയിലെ ഏറ്റവും കുറഞ്ഞ മൂല്യമുള്ള അംഗത്തെ കണ്ടെത്തി ആദ്യ സ്ഥാനത്തേക്ക് മാറ്റിക്കൊണ്ടാണ് സെലക്ഷൻ സോർട്ട് ആരംഭിക്കുന്നത്. അതേസമയം ആദ്യ സ്ഥാനത്തെ അംഗത്തെ ചെറിയ മൂല്യമുള്ള അംഗത്തിന്റെ സ്ഥാനത്തേക്കും മാറ്റുന്നു. അതിനുശേഷം രണ്ടാമത്തെ കുറഞ്ഞ മൂല്യമുള്ള അംഗത്തെ കണ്ടെത്തി രണ്ടാം സ്ഥാനത്തുള്ള അംഗവുമായി പരസ്പരം മാറ്റം ചെയ്യുന്നു. അങ്ങനെ എല്ലാ അംഗങ്ങളെയും ക്രമീകരിക്കപ്പെടുന്നതുവരെ ഈ പ്രവർത്തനം തുടരുന്നു. ഒരു അറെയിലെ കുറഞ്ഞ മൂല്യമുള്ള അംഗത്തെ കണ്ടെത്തി അനുയോജ്യമായ സ്ഥാനത്തെ അംഗവുമായി മാറ്റം ചെയ്യുന്ന പ്രക്രിയയെ സ്ഥാനമാറ്റം എന്ന് പറയുന്നു. N അംഗങ്ങളുള്ള ഒരു സെലക്ഷൻ സോർട്ടിൽ N-1 സ്ഥാനമാറ്റങ്ങൾ ഉണ്ടായിരിക്കും. ഉദാഹരണത്തിന് താഴെ നൽകിയിരിക്കുന്ന സംഖ്യകളുടെ പട്ടിക നോക്കുക.

പ്രാരംഭ പട്ടിക

32	23	10	2	30

സ്ഥാനമാറ്റം 1

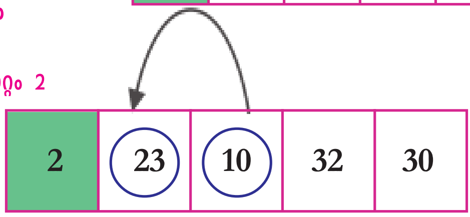


സ്ഥാനമാറ്റം 1: പട്ടികയിൽ നിന്നും ഏറ്റവും ചെറിയ അംഗമായ 2 തിരഞ്ഞെടുക്കുന്നു, അത് ആദ്യത്തെ അംഗവുമായി സ്ഥാനമാറ്റം ചെയ്യപ്പെടുന്നു.

സ്ഥാനമാറ്റം 1 നുശേഷം

2	23	10	32	30

സ്ഥാനമാറ്റം 2

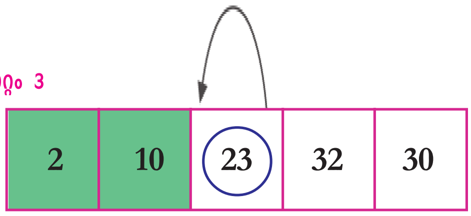


സ്ഥാനമാറ്റം 2: പട്ടികയിൽ നിന്നും 2 ഒഴികെയുള്ളതിൽ ചെറിയ അംഗമായ 10 തിരഞ്ഞെടുക്കുന്നു, അത് രണ്ടാമത്തെ അംഗവുമായി സ്ഥാനമാറ്റം ചെയ്യപ്പെടുന്നു.

സ്ഥാനമാറ്റം 2 നുശേഷം

2	10	23	32	30

സ്ഥാനമാറ്റം 3

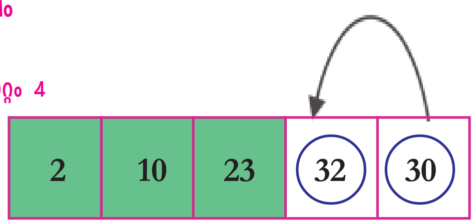


സ്ഥാനമാറ്റം 3: പട്ടികയിൽ നിന്നും 2, 10 എന്നിവ ഒഴികെയുള്ളതിൽ ചെറിയ അംഗമായ 23 തിരഞ്ഞെടുക്കുന്നു, അത് മൂന്നാമത്തെ അംഗവുമായി സ്ഥാനമാറ്റം ചെയ്യപ്പെടുന്നു.

സ്ഥാനമാറ്റം 3 നുശേഷം

2	10	23	32	30

സ്ഥാനമാറ്റം 4



സ്ഥാനമാറ്റം 4: പട്ടികയിൽ നിന്നും 2, 10, 23 എന്നിവ ഒഴികെയുള്ളതിൽ ചെറിയ അംഗമായ 30 തിരഞ്ഞെടുക്കുന്നു, അത് നാലാമത്തെ അംഗവുമായി സ്ഥാനമാറ്റം ചെയ്യപ്പെടുന്നു.

സ്ഥാനമാറ്റം 4 നുശേഷം

2	10	23	30	32

ഓരോ തവണയും ഒരു സ്ഥാനമാറ്റം ഉദ്ദേശിച്ചിട്ടുണ്ടെങ്കിലും, ഏറ്റവും കുറഞ്ഞ മൂല്യം ശരിയായ സ്ഥലത്ത് ആണെങ്കിൽ സ്ഥാനമാറ്റം സംഭവിക്കുന്നില്ല. സ്ഥാനമാറ്റം മൂന്നിൽ ഇത് നിങ്ങൾക്ക് കാണുവാൻ സാധിക്കും

സെലക്ഷൻ സോർട്ടിന്റെ അൽഗോരിതം

1. ആരംഭിക്കുക
2. N ന്റെ വില സ്വീകരിക്കുക
3. I ← 0
4. ഘട്ടങ്ങൾ 5, 6 എന്നിവ I ← N-1 ആകുന്നതുവരെ ആവർത്തിക്കുക
5. AR[I] ലേയ്ക്ക് ഡാറ്റ സ്വീകരിക്കുക
6. I ← I +1
7. I ← 0
8. ഘട്ടങ്ങൾ 9 മുതൽ 14 വരെ I ← N-1 ആകുന്നതുവരെ ആവർത്തിക്കുക
9. MIN ← AR[I], POS ← I
10. J ← I +1 മുതൽ N-1 ആകുന്നതുവരെ ഘട്ടം 11, 12 ആവർത്തിക്കുക
11. IF AR[J] < MIN ആണെങ്കിൽ MIN ← AR[J], POS ← J
12. J ←J +1
13. IF POS <> I ആണെങ്കിൽ AR[POS] ← AR[I], AR[I] ← MIN
14. I ← I +1
15. I ← 0
16. ഘട്ടങ്ങൾ 15, 16 എന്നിവ I ← N-1 ആകുന്നതുവരെ ആവർത്തിക്കുക
17. AR[I] പ്രദർശിപ്പിക്കുക.
18. I ← I +1
19. അവസാനിപ്പിക്കുക

പ്രോഗ്രാം 8.3: ആരോഹണ ക്രമത്തിൽ അംഗങ്ങളെ ക്രമീകരിക്കുന്നതിനുള്ള സെലക്ഷൻ സോർട്ട്

```
#include <iostream>
using namespace std;
int main()
{   int AR[25], N, I, J, MIN, POS;
    cout<<"How many elements? ";
    cin>>N;
    cout<<"Enter the array elements: ";
    for(I=0; I<N; I++)
    cin>>AR[I];
    for(I=0; I < N-1; I++)
    {
        MIN=AR[I];
        POS=I;
```

```
        for(J = I+1; J < N; J++)
            if(AR[J]<MIN)
            {
                MIN=AR[J];
                POS=J;
            }
        if(POS != I)
        {
            AR[POS]=AR[I];
            AR[I]=MIN;
        }
    }
    cout<<"Sorted array is: ";
    for(I=0; I<N; I++)
        cout<<AR[I]<<"\t";
    return 0;
}
```

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
How many elements? 5
Enter the array elements: 12 3   6   1   8
Sorted array is: 1    3    6    8    12
```

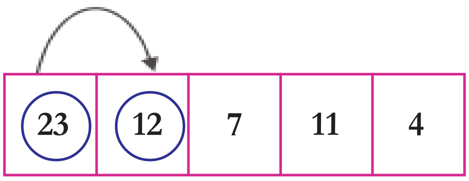
b. ബബിൾ സോർട്ട് (Bubble sort)

ബബിൾ സോർട്ട് അൽഗോരിതം ക്രമീകരിക്കേണ്ട അറെയിലെ അടുത്തടുത്ത ഓരോ ജോഡി അംഗങ്ങളെ താരതമ്യം ചെയ്യുകയും അവ തെറ്റായ ക്രമത്തിലാണെങ്കിൽ പരസ്പരം സ്ഥാനമാറ്റം നടത്തുകയും ചെയ്യുന്നു. ഒരു സ്ഥാനമാറ്റവും ആവശ്യമില്ലാത്തതുവരെ ഈ പ്രക്രിയ ആവർത്തിക്കപ്പെടുന്നു, ഈ അവസ്ഥയിൽ അറെ ക്രമീകരിക്കപ്പെട്ടതായി കരുതാം. ഒരു ഉദാഹരണത്തിന്റെ സഹായത്തോടെ ഈ പ്രക്രിയ പരിശോധിക്കാം.

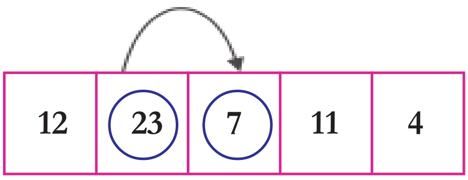
പ്രാരംഭ പട്ടിക

23	12	7	11	4

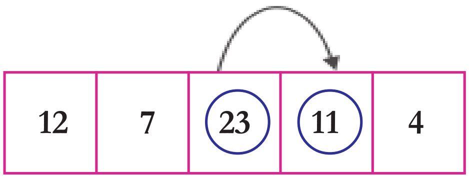
സ്ഥാനമാറ്റം 1



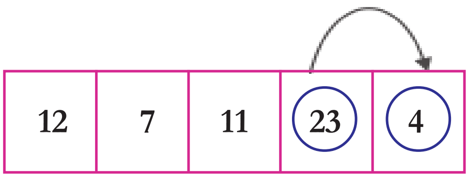
ആദ്യത്തെ ആദ്യ രണ്ട് അംഗങ്ങളായ 23, 12 എന്നിവ താരതമ്യം ചെയ്ത ശേഷം പരസ്പരം സ്ഥാനമാറ്റം ചെയ്യുന്നു.



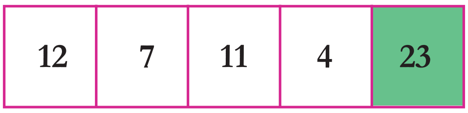
പരിഷ്കരിച്ച പട്ടികയിലെ രണ്ടാമത്തെയും മൂന്നാമത്തെയും അംഗങ്ങളായ 23, 7 എന്നിവ താരതമ്യം ചെയ്ത ശേഷം പരസ്പരം സ്ഥാനമാറ്റം ചെയ്യുന്നു.



പരിഷ്കരിച്ച പട്ടികയിലെ മൂന്നാമത്തെയും നാലാമത്തെയും അംഗങ്ങളായ 23, 11 എന്നിവ താരതമ്യം ചെയ്ത ശേഷം പരസ്പരം സ്ഥാനമാറ്റം ചെയ്യുന്നു.

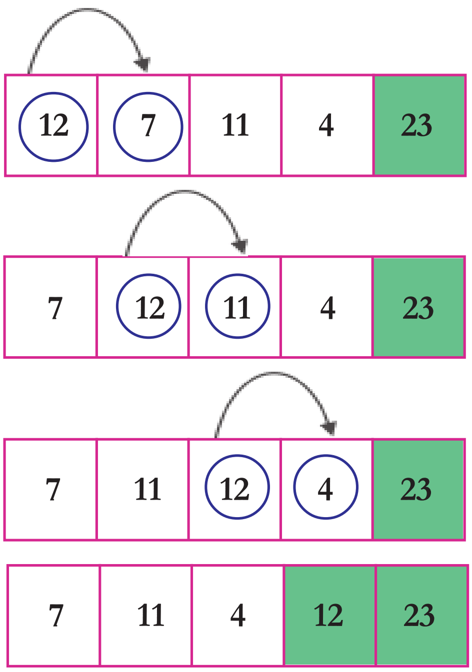


പരിഷ്കരിച്ച പട്ടികയിലെ നാലാമത്തെയും അഞ്ചാമത്തെയും അംഗങ്ങളായ 23, 4 എന്നിവ താരതമ്യം ചെയ്ത ശേഷം പരസ്പരം സ്ഥാനമാറ്റം ചെയ്യുന്നു.



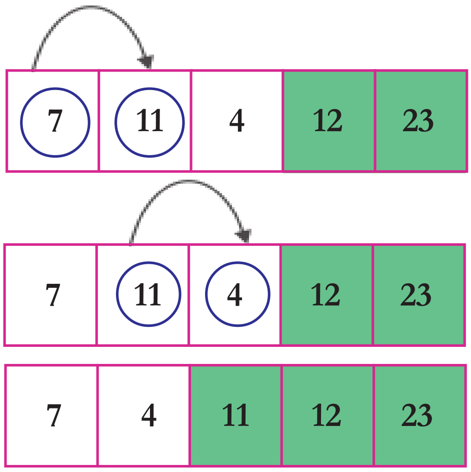
ആദ്യത്തെ സ്ഥാനമാറ്റം കഴിയുമ്പോൾ അറെയിലെ ഏറ്റവും വലിയ അംഗമായ 23 അറെയുടെ അവസാന സ്ഥാനത്ത് എത്തുന്നു.

സ്ഥാനമാറ്റം 2



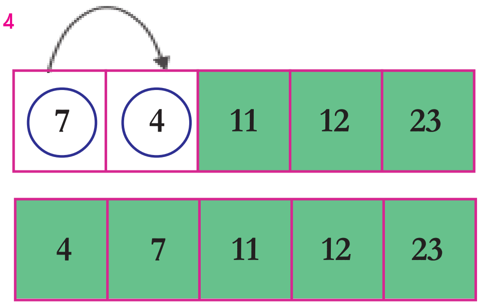
രണ്ടാമത്തെ സ്ഥാനമാറ്റത്തിൽ ആദ്യത്തെ നാല് അംഗങ്ങളെ മാത്രം പരിഗണിക്കുന്നു. ഒന്നാമത്തെ സ്ഥാനമാറ്റത്തിലെ അതേ പ്രക്രിയ തുടരുന്നു, അതിന്റെ ഫലമായി രണ്ടാമത്തെ സ്ഥാനമാറ്റത്തിൻറെ അവസാനം രണ്ടാമത്തെ വലിയ സംഖ്യയായ 12 അറെയുടെ നാലാം സ്ഥാനത്ത് എത്തുന്നു.

സ്ഥാനമാറ്റം 3



മൂന്നാമത്തെ സ്ഥാനമാറ്റത്തിൽ, 23 ഉം 12 ഉം ഒഴികെ പട്ടികയിലെ മൂന്ന് അംഗങ്ങളെ മാത്രമേ പരിഗണിക്കുന്നുള്ളൂ. മേൽപ്പറഞ്ഞ സ്ഥാനമാറ്റത്തിലെ അതേ പ്രക്രിയ തുടരുന്നു, അതിന്റെ ഫലമായി മൂന്നാമത്തെ സ്ഥാനമാറ്റത്തിന്റെ അവസാനം 11 എന്ന അംഗം അറെയുടെ മൂന്നാം സ്ഥാനത്ത് എത്തുന്നു.

സ്ഥാനമാറ്റം 4



നാലാമത്തെ സ്ഥാനമാറ്റത്തിൽ, 23, 12, 11 എന്നിവ ഒഴികെ പട്ടികയിലെ രണ്ട് അംഗങ്ങളെ മാത്രമേ പരിഗണിക്കുന്നുള്ളൂ. മേൽപ്പറഞ്ഞ സ്ഥാനമാറ്റത്തിലെ അതേ പ്രക്രിയ തുടരുന്നു, അതിന്റെ ഫലമായി നാലാമത്തെ സ്ഥാനമാറ്റത്തിന്റെ A അവസാനം 7 എന്ന അംഗം അറെയുടെ രണ്ടാം സ്ഥാനത്തെത്തുന്നു. 4 ഒന്നാം സ്ഥാനത്തും എത്തുന്നു

N അംഗങ്ങളുള്ള ഒരു ബബിൾ സോർട്ടിൽ N-1 സ്ഥാനമാറ്റങ്ങൾ ഉണ്ടായിരിക്കും. ഓരോ സ്ഥാനമാറ്റത്തിലും പരിഷ്കരിച്ച അറെയിലെ പരിഗണിക്കപ്പെടുന്ന അംഗങ്ങളുടെ എണ്ണം ഒന്നു വീതം കുറയും.

ബബിൾ സോർട്ടിന്റെ അൽഗോരിതം

1. ആരംഭിക്കുക
2. N ന്റെ വില സ്വീകരിക്കുക
3. I ← 0
4. ഘട്ടങ്ങൾ 5, 6 എന്നിവ I ← N-1 ആകുന്നതുവരെ ആവർത്തിക്കുക
5. AR[I] ലേയ്ക്ക് ഡാറ്റ സ്വീകരിക്കുക
6. I ← I +1
7. I ← 1
8. ഘട്ടങ്ങൾ 9, 12 എന്നിവ I ← N-1 ആകുന്നതുവരെ ആവർത്തിക്കുക
9. J ← 0 മുതൽ N-2 ആകുന്നതുവരെ ഘട്ടം 10, 11 എന്നിവ ആവർത്തിക്കുക
10. IF AR[J] > AR[J+1] ആണെങ്കിൽ TEMP←AR[J], AR[J] ←AR[J+1], AR[J+1] ← MIN

11. J ← J +1
12. I ← I +1
13. I ← 0
14. ഘട്ടങ്ങൾ 14, 15 എന്നിവ I ← N-1 ആകുന്നതുവരെ ആവർത്തിക്കുക
15. AR[I] പ്രദർശിപ്പിക്കുക.
16. I ← I +1
17. അവസാനിപ്പിക്കുക

പ്രോഗ്രാം 8.4: ആരോഹണ ക്രമത്തിൽ അംഗങ്ങളെ ക്രമീകരിക്കുന്നതിനുള്ള ബബിൾ സോർട്ട്

```
#include <iostream>
using namespace std;
int main()
{   int AR[25],N;
    int I, J, TEMP;
    cout<<"How many elements? ";
    cin>>N;
    cout<<"Enter the array elements: ";
    for(I=0; I<N; I++)
        cin>>AR[I];
    for(I=1; I<N; I++)
        for(J=0; J<N-I; J++)
            if(AR[J] > AR[J+1])
            {
                TEMP = AR[J];
                AR[J] = AR[J+1];
                AR[J+1] = TEMP;
            }
    cout<<"Sorted array is: ";
    for(I=0; I<N; I++)
        cout<<AR[I]<<"\t";
}
```

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
How many elements? 5
Enter the array elements: 23  10  -3  7   11
Sorted array is: -3    7    10    11    23
```

8.2.3 തിരയൽ (Searching)

അറെയിലെ ഒരു അംഗത്തിന്റെ സ്ഥാനം കണ്ടുപിടിക്കുന്ന പ്രക്രിയയെ തിരയൽ (Searching) എന്നു പറയുന്നു. തന്നിരിക്കുന്ന ഡാറ്റയെ അറെയിൽ കണ്ടെത്തിയാൽ ആ തിരയൽ വിജയിച്ചു എന്നു പറയാം, അതായത് തന്നിരിക്കുന്ന ഡാറ്റ അറെയിലെ ഒരു അംഗമാണ്. അല്ലെങ്കിൽ തിരയൽ പരാജയപ്പെട്ടു. തിരയലിന് രേഖീയ തിരയൽ, ബൈനറി തിരയൽ

എന്നിങ്ങനെ രണ്ട് സമീപന രീതികൾ ഉണ്ട്. തിരയലിന് തിരഞ്ഞെടുക്കുന്ന അൽഗോരിതം അറെയിലെ അംഗങ്ങളുടെ ക്രമീകരണത്തെ ആശ്രയിച്ചിരിക്കുന്നു ക്രമരഹിതമായി അംഗങ്ങളെ ക്രമീകരിച്ചിരിക്കുന്ന അറെയിൽ രേഖീയ തിരയൽ രീതി ഉപയോഗിക്കുന്നു, എന്നാൽ അംഗങ്ങളെ ആരോഹണ ക്രമത്തിലോ അവരോഹണ ക്രമത്തിലോ വിന്യസിച്ചിരിക്കുന്ന അറെയിൽ ബൈനറി തിരയൽ രീതി ഉപയോഗിക്കുന്നതാണ് അഭികാമ്യം. ഈ തിരയൽ രീതികൾ താഴെ വിവരിക്കുന്നു.

a. രേഖീയ തിരയൽ (Linear search)

അറെയിൽ ഒരു പ്രത്യേക ഡാറ്റ കണ്ടെത്തുവാനുള്ള ഒരു രീതിയാണ് രേഖീയ തിരയൽ. രേഖീയ തിരയൽ പട്ടികയിലെ ഒന്നാമത്തെ അംഗത്തിൽ നിന്നും ആരംഭിച്ച് ക്രമമനുസരിച്ച്, ഓരോ അംഗത്തേയും പരിശോധിക്കുന്നു. ഈ പരിശോധന ഒന്നുകിൽ അംഗത്തെ കണ്ടെത്തുന്നതു വരെ അല്ലെങ്കിൽ പട്ടികയുടെ അവസാനം വരെ തുടരുന്നു.

50, 18, 48, 35, 45, 26, 12 എന്നീ അംഗങ്ങളുള്ള ഒരു അറെയിൽ നിന്ന് '45' എന്ന അംഗത്തെ തിരയണമെന്ന് കരുതുക. ആദ്യ അംഗമായ 50 ൽ നിന്നും രേഖീയ തിരയൽ ആരംഭിക്കുന്നു, അത് ഓരോ അംഗത്തെയും താരതമ്യം ചെയ്യുന്നു അഞ്ചാം സ്ഥാനത്ത് എത്തുമ്പോൾ ചിത്രം 8.3 ൽ കാണിച്ചിരിക്കുന്നത് പോലെ 45 കണ്ടെത്തുന്നു.

ഇൻഡക്സ്	പട്ടിക	താരതമ്യം
0	50	50 == 45 : തെറ്റ്
1	18	18 == 45 : തെറ്റ്
2	48	48 == 45 : തെറ്റ്
3	35	35 ==45 : തെറ്റ്
4	45	45 ==45 : ശരി
5	26	
6	12	

ചിത്രം 8.3 രേഖീയ തിരയൽ

രേഖീയ തിരയലിന്റെ അൽഗോരിതം

1. ആരംഭിക്കുക
2. N ← ന്റെ വില സ്വീകരിക്കുക
3. I ← 0
4. ഘട്ടങ്ങൾ 5, 6 എന്നിവ I ← N-1 ആകുന്നതുവരെ ആവർത്തിക്കുക
5. AR[I] ലേയ്ക്ക് ഡാറ്റ സ്വീകരിക്കുക
6. I ← I +1
7. ITEM ന്റെ വില സ്വീകരിക്കുക
8. I ← 0, LOC ← 0

9. ഘട്ടങ്ങൾ 10, 11 എന്നിവ I ← N-1 ആകുന്നതുവരെ ആവർത്തിക്കുക
10. IF AR[I] = ITEM ആണെങ്കിൽ LOC ←1, ഘട്ടം 12 ൽ പോകുക
11. I ← I +1
12. IF LOC = 1 ആണെങ്കിൽ തിരയൽ വിജയിച്ചു എന്ന് പ്രദർശിപ്പിക്കുക അല്ലെങ്കിൽ തിരയൽ പരാജയപ്പെട്ടു എന്നും പ്രദർശിപ്പിക്കുക.
13. അവസാനിപ്പിക്കുക

പ്രോഗ്രാം 8.5: അറെയിലുള്ള ഒരു അംഗത്തെ കണ്ടെത്തുന്നതിനുള്ള രേഖീയ തിരയൽ

```
#include <iostream>
using namespace std;
int main()
{
    int AR[25], N;
    int I, ITEM, LOC=-1;
    cout<<"How many elements? ";
    cin>>N;
    cout<<"Enter the array elements: ";
    for(I=0; I<n; I++)
      cin>>AR[I];
    cout<<"Enter the item you are searching for: ";
    cin>>ITEM;
    for(I=0; I<N; I++)
        if(AR[I] == ITEM)
        {
            LOC=I;
            break;
        }
    if(LOC!=-1)
        cout<<"The item is found at position "<<LOC+1;
    else
        cout<<"The item is not found in the array";
    return 0;
}
```

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
How many Elements?  7
Enter the array elements: 12 18 26 35 45 48 50
Enter the item you are searching for: 35
The item is found at position 4
```

```
How many Elements? 7
Enter the array elements: 12 18 26 35 45 48 50
Enter the item you are searching for: 25
The item is not found in the array
```

പട്ടികയുടെ മുൻപന്തിയിലാണ് തിരയേണ്ട അംഗമെങ്കിൽ ഏതാനും താരതമ്യങ്ങൾ കൊണ്ട് രേഖീയ തിരയൽ പ്രക്രിയ അവസാനിക്കും. പട്ടികയുടെ അവസാന ഭാഗത്താണ് തിരയേണ്ട അംഗമെങ്കിൽ തിരയൽ പ്രക്രിയയിലെ താരതമ്യങ്ങളുടെ എണ്ണം വളരെ വലുതായിരിക്കും, ഉദാഹരണത്തിന് 10000 അംഗങ്ങളുള്ള ഒരു പട്ടികയിൽ പരമാവധി താരതമ്യങ്ങളുടെ എണ്ണം 10000 ആയിരിക്കും.

b. ബൈനറി തിരയൽ (Binary Search)

നമ്മൾ വിശകലനം ചെയ്ത രേഖീയ തിരയൽ അൽഗോരിതം ലളിതവും കുറച്ച് അംഗങ്ങളുള്ള അറേകൾക്ക് യോജിച്ചതുമാണ്. എന്നാൽ അറയിൽ നിരവധി അംഗങ്ങൾ ഉണ്ടെങ്കിൽ ധാരാളം തിരയലുകൾ ആവശ്യമായി വരും. ഈ സാഹചര്യത്തിൽ കൂടുതൽ കാര്യക്ഷമമായ ഒരു അൽഗോരിതം ഉപയോഗിക്കേണ്ടതുണ്ട്. അറെയിലെ അംഗങ്ങളെ ആരോഹണ ക്രമത്തിലോ അവരോഹണ ക്രമത്തിലോ ക്രമീകരിച്ചിട്ടുണ്ടെങ്കിൽ തിരച്ചിൽ സമയം കുറയ്ക്കാൻ കഴിയുന്ന കൂടുതൽ മെച്ചപ്പെട്ട ബൈനറി തിരയൽ അൽഗോരിതം നമുക്ക് ഉപയോഗിക്കാൻ കഴിയും

ഉദാഹരണത്തിന്, ഒരു നിഘണ്ടുവിൽ 'മോഡം' എന്ന പദത്തിന്റെ അർത്ഥം കണ്ടെത്തണമെന്ന് കരുതുക. തീർച്ചയായും നമ്മൾ ഒന്നാമത്തെ പേജിന്റെ ആദ്യ വാക്കു മുതൽ തിരച്ചിൽ ആരംഭിക്കുകയില്ല, മറിച്ച് നമ്മൾ തിരയുന്ന വാക്ക് ഏതാണ് ഉദ്ദേശം വെച്ച് നിഘണ്ടു തുറക്കുന്നു. നമുക്ക് തിരയേണ്ട വാക്ക് ആ പുറത്ത് ഇല്ലെങ്കിൽ പിന്നീടുള്ള തിരച്ചിൽ ഒരു പകുതി അവഗണിച്ച് മറ്റെ പകുതിയിൽ തിരയുന്നു. തിരച്ചിൽ നടത്തി ആവശ്യമായ പദം കണ്ടെത്തുവരെ അല്ലെങ്കിൽ നിഘണ്ടുവിൽ ഈ പദം ഇല്ല എന്ന് ഉറപ്പാക്കുക്കുന്നതുവരെ ഈ പ്രക്രിയ തുടർന്നുകൊണ്ടേയിരിക്കും. ഈ തിരച്ചിൽ രീതി ഒരു നിഘണ്ടുവിൽ സാധ്യമാണ്, കാരണം വാക്കുകൾ അക്ഷരമാലയുടെ ആരോഹണക്രമത്തിലാണ് അവിടെ ക്രമീകരിച്ചിരിക്കുന്നത്.

ബൈനറി തിരയൽ അൽഗോരിതം കുറഞ്ഞ തിരച്ചിലുകൾ കൊണ്ട് പട്ടികയിൽനിന്നും ഒരു അംഗത്തിൻറെ സ്ഥാനം കണ്ടെത്തുന്നു.

ബൈനറി തിരയലിന്റെ അൽഗോരിതം

1. ആരംഭിക്കുക
2. MAX ന്റെ വില സ്വീകരിക്കുക
3. I ← 0
4. ഘട്ടങ്ങൾ 5, 6 എന്നിവ I = MAX -1 ആകുന്നതുവരെ ആവർത്തിക്കുക
5. LIST [I] ലേയ്ക്ക് ഡാറ്റ സ്വീകരിക്കുക
6. I I +1
7. ITEM ന്റെ വില സ്വീകരിക്കുക

8. FIRST ← 0, LAST ← MAX -1
9. FIRST = LAST ആകുന്നതുവരെ ഘട്ടങ്ങൾ 10 മുതൽ 12 വരെ ആവർത്തിക്കുക
10. MIDDLE ← (FIRST + LAST) / 2
11. IF LIST [MIDDLE] = ITEM ആണെങ്കിൽ LOC ← 1, ഘട്ടം 13 ൽ പോകുക
12. IF ITEM < LIST [MIDDLE] ആണെങ്കിൽ LAST ← MIDDLE-1 അല്ലെങ്കിൽ FIRST ← MIDDLE+1
13. IF LOC = 1 ആണെങ്കിൽ തിരയൽ വിജയിച്ചു എന്ന് പ്രദർശിപ്പിക്കുക അല്ലെങ്കിൽ തിരയൽ പരാജയപ്പെട്ടു എന്നും പ്രദർശിപ്പിക്കുക.
14. അവസാനിപ്പിക്കുക

പ്രോഗ്രാം 8.6: അറെയിലുള്ള ഒരു അംഗം കണ്ടെത്തുന്നതിനുള്ള ബൈനറി തിരയൽ

```
#include <iostream>
using namespace std;
int main()
{   int LIST[25],MAX;
    int FIRST, LAST, MIDDLE, I, ITEM, LOC=-1;
    cout<<"How many elements? ";
    cin>>MAX;
    cout<<"Enter array elements in ascending order: ";
    for(I=0; I<MAX; I++)
        cin>>LIST[I];
    cout<<"Enter the item to be searched: ";
    cin>>ITEM;
    FIRST=0;
    LAST=MAX-1;
    while(FIRST<=LAST)
    {
        MIDDLE=(FIRST+LAST)/2;
        if(ITEM == LIST[MIDDLE])
        {
            LOC = MIDDLE;
            break;
        }
        if(ITEM < LIST[MIDDLE])
            LAST = MIDDLE-1;
        else
            FIRST = MIDDLE+1;
    }
    if(LOC != -1)
```

```
        cout<<"The item is found at position "<<LOC+1;
    else
        cout<<"The item is not found in the array";
    return 0;
}
```

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
How many elements? 7
Enter array elements in ascending order: 21 28 33 35 45 58 61
Enter the item to be searched: 35
The item is found at position 4
```

ബൈനറി തിരയൽ പ്രവർത്തനം വ്യക്തമാക്കുന്നതിന് താഴെപ്പറയുന്ന 7 (MAX=7) അംഗങ്ങളുള്ള അറെ പരിഗണിക്കാം, തിരയേണ്ടുന്ന അംഗം 45 ആണെന്നും കരുതുക.

0	1	2	3	4	5	6
21	28	33	35	45	58	61

MAX = 7

FIRST = 0

LAST = 6

FIRST<=LAST,

ആയതിനാൽ നമുക്ക് പ്രവർത്തനം ആരംഭിക്കാം

0	1	2	3	4	5	6
21	28	33	35	45	58	61

MIDDLE = (**FIRST**+**LAST**)/2 = (0+6)/2 = 3
ഇവിടെ **LIST[MIDDLE]** അതായത്, **LIST[3]** ന്റെ വിലയും **45** ഉം തുല്യമല്ല, എന്നാൽ **LIST[3]** ന്റെ വില തിരയൽ വിലയേക്കാൾ കുറവാണ്. അതിനാൽ
FIRST = **MIDDLE** + 1 = 3 + 1 = 4, **LAST** = 6

FIRST<=LAST,

ആയതിനാൽ അടുത്ത തിരച്ചിൽ ആരംഭിക്കാം

0	1	2	3	4	5	6
21	28	33	35	45	58	61

MIDDLE = (**FIRST**+**LAST**)/2 = (4+6)/2= 5
ഇവിടെ **LIST[MIDDLE]** അതായത്, **LIST[5]** ന്റെ വിലയും **45** ഉം തുല്യമല്ല, എന്നാൽ **LIST[5]** ന്റെ വില തിരയൽ വിലയേക്കാൾ വലുതാണ്. അതിനാൽ
FIRST = 4, **LAST** = **MIDDLE** - 1=5 -1 = 4,

FIRST<=LAST,

ആയതിനാൽ അടുത്ത തിരച്ചിൽ ആരംഭിക്കാം

0	1	2	3	4	5	6
21	28	33	35	45	58	61

MIDDLE = (**FIRST**+**LAST**)/2 = (4+4)/2 = 4
ഇവിടെ **LIST[MIDDLE]** അതായത്, **LIST[4]** ന്റെ വിലയും **45** ഉം തുല്യമാണ്, കൂടാതെ തിരയൽ വിജയകരമായി അവസാനിച്ചിരിക്കുന്നു.

ബൈനറി സെർച്ചിൽ, 100,00,00,000 (100 കോടി) അംഗങ്ങളുള്ള ഒരു അറെയിൽ ഒരു അംഗം തിരയാൻ പരമാവധി 30 താരതമ്യങ്ങൾ ആവശ്യമാണ്. അറെയിലെ അംഗങ്ങളുടെ എണ്ണം ഇരട്ടിയായെങ്കിൽ, ഒരു താരതമ്യം മാത്രമേ കൂടുതലായി ആവശ്യമുള്ളു.

പട്ടിക 8.1 ബൈനറി തിരയലും രേഖീയ തിരയലും തമ്മിലുള്ള വ്യത്യാസം കാണിച്ചിരിക്കുന്നു:

രേഖീയ തിരയൽ	ബൈനറി തിരയൽ
• അംഗങ്ങൾ ഏതെങ്കിലും രീതിയിൽ ക്രമീകരിക്കേണ്ടതില്ല • തിരയൽ പ്രവർത്തനത്തിന് കൂടു തൽ സമയം എടുക്കുന്നു • എല്ലാ അംഗങ്ങളേയും സന്ദർശിക്കേണ്ടതായി വരാം • കുറച്ച് അംഗങ്ങളുള്ള അറെകൾക്ക് അനുയോജ്യം.	• അംഗങ്ങളെ ആരോഹണ ക്രമത്തിലോ അവരോഹണ ക്രമത്തിലോ ക്രമീകരിച്ചിരിക്കണം • തിരയൽ പ്രവർത്തനത്തിന് വളരെ കുറച്ച് സമയമേ ആവശ്യമുള്ളു • എല്ലാ അംഗങ്ങളേയും സന്ദർശിക്കേണ്ടതില്ല • കൂടുതൽ അംഗങ്ങളുള്ള അറെകൾക്ക് അനുയോജ്യം

പട്ടിക 8.1 ബൈനറി സെർച്ചും രേഖീയ സെർച്ചും തമ്മിലുള്ള താരതമ്യം

8.3 ദ്വിമാന അറെകൾ (Two dimensional Arrays)

50 വിദ്യാർത്ഥികളുടെ ആറു വ്യത്യസ്ത വിഷയങ്ങളിലെ മാർക്കുകൾ നമുക്ക് സൂക്ഷിക്കേണ്ടതുണ്ട് എന്ന് കരുതുക. ഇവിടെ നമുക്ക് 50 അംഗങ്ങൾ ഉള്ള 6 ഏകമാന അറെകൾ ഉപയോഗിക്കാം. എന്നാൽ ഈ ക്രമീകരണം കൈകാര്യം ചെയ്യുന്നത് എളുപ്പമുള്ള കാര്യമല്ല. ഈ സാഹചര്യത്തിൽ നമുക്ക് അറെകളുടെ അറെ അല്ലെങ്കിൽ ദ്വിമാന അറെ ഉപയോഗിക്കാം.

ഒരു ദ്വിമാന അറെയിലെ ഓരോ അംഗവും ഒരു അറെയാണ്. ഉദാഹരണമായി, AR[m][n] എന്ന ദ്വിമാന അറെയിൽ n അംഗങ്ങളുള്ള m അറെകൾ ഉണ്ട്. അല്ലെങ്കിൽ m വരികളും n നിരകളും അടങ്ങുന്ന ഒരു പട്ടികയാണ് AR [m][n] എന്ന ദ്വിമാന അറെ.

8.3.1 ദ്വിമാന അറെകളുടെ പ്രഖ്യാപനം (Declaring 2D Array)

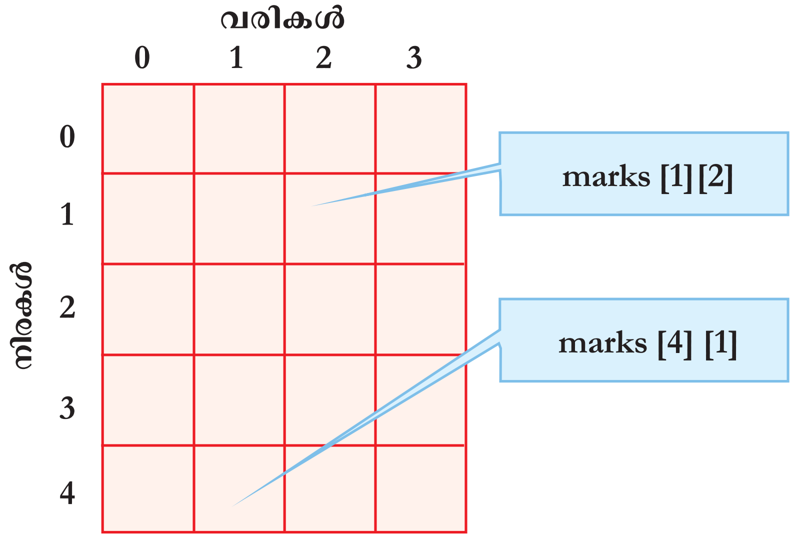
C ++ ലെ ദ്വിമാന അറെ നീക്കിവെയ്ക്കലിന്റെ വാക്യഘടന താഴെ കൊടുക്കുന്നു

```
data_type  array_name[rows][columns];
```

വാക്യഘടനയിൽ `data_type` എന്നത് അറെയിലെ അംഗങ്ങളുടെ ഡേറ്റയുടെ ഇനമാണ് സൂചിപ്പിക്കുന്നത്. `array_name` എന്നത് അറെയുടെ പേരും `rows` എന്നത് ദ്വിമാന അറെയിലെ വരികളുടെ എണ്ണവും `columns` എന്നത് ദ്വിമാന അറെയിലെ നിരകളുടെ എണ്ണവും സൂചിപ്പിക്കുന്നു. വരികളുടെയും നിരകളുടെയും സൂചിക 0-ൽ ആരംഭിച്ച് യഥാക്രമം വരികൾ rows -1 ലും നിരകൾ coulmns - 1 ലും അവസാനിക്കും. 5 വരികളും 4 നിരകളും ഉള്ള ഒരു ദ്വിമാന അറെ പ്രഖ്യാപനത്തിന്റെ ഉദാഹരണം താഴെ കൊടുക്കുന്നു.

```
int marks[5][4];
```

ചിത്രം 8.4 ൽ കാണിച്ചിരിക്കുന്നത് പോലെ. ഈ അറെയിലെ അംഗങ്ങൾ `marks[0][0]`, `marks[0][1]`, `marks[0][2]`, `marks[0][3]`, `marks[1][0]`, `marks[1][1]`, ..., `marks[4][3]` എന്നിവയാകുന്നു.



ചിത്രം 8.4 ഒരു ദ്വിമാന അറെയുടെ ഘടന

ഒരു ദ്വിമാന അറേക്ക് ആവശ്യമായ മെമ്മറിയുടെ അളവ് അതിന്റെ ഡാറ്റ ഇനം, വരികളുടെ എണ്ണം, നിരകളുടെ എണ്ണം എന്നിവയുടെ അടിസ്ഥാനത്തിലാണ് കണക്കാക്കുന്നത്. ദ്വിമാന അറേക്ക് ആവശ്യമായ ആകെ ബൈറ്റുകളുടെ എണ്ണം കണക്കുകൂട്ടുന്നതിനുള്ള സൂത്രവാക്യം താഴെ കൊടുത്തിരിക്കുന്നു.

ആകെ ബൈറ്റുകൾ = `sizeof`(ഡാറ്റ ഇനം) × വരികളുടെ എണ്ണം × നിരകളുടെ എണ്ണം

ഉദാഹരണത്തിന്, മുകളിൽ പറഞ്ഞിരിക്കുന്ന `marks[5][4]` ന് $4 \times 5 \times 4 = 80$ ബൈറ്റ് മെമ്മറി ആവശ്യമാണ്.

8.3.2 മെട്രിക്സായി ദ്വിമാന അറേകൾ (Matrices as 2D arrays)

ഗണിതശാസ്ത്രത്തിലെ ഉപയോഗപ്രദമായ ഒരു ആശയമാണ് മെട്രിക്സ്. ഒരു മെട്രിക്സ് എന്നത് m വരികളിലും n നിരകളിലുമായി ഒരു പട്ടികയുടെ രൂപത്തിൽ ക്രമീകരിച്ചിരിക്കുന്ന m × n സംഖ്യകളുടെ ഒരു ഗണമാണെന്ന് നമുക്കറിയാം. ദ്വിമാന അറെയുടെ സഹായത്തോടെ മെട്രിക്സ് പ്രതിനിധീകരിക്കാനാകും. ഒരു ദ്വിമാന അറെ പ്രോസസ് ചെയ്യുന്നതിന് നിങ്ങൾ നെസ്റ്റഡ് ലൂപ്പ് ഉപയോഗിക്കണം. ഒരു ലൂപ്പ് വരികളേയും അടുത്തത് നിരകളേയും പ്രോസസ്സുചെയ്യുന്നു. സാധാരണയായി പുറത്തെ ലൂപ്പ് വരികൾക്കും അകത്തെ ലൂപ്പ് നിരകൾക്കും വേണ്ടിയുള്ളതാണ്. പ്രോഗ്രാം 8.7 ഉപയോഗിച്ച് m വരികളും n നിരകളും ഉള്ള ഒരു മെട്രിക്സ് പ്രോസസ്സ് ചെയ്യുന്നു.

പ്രോഗ്രാം 8.7. m വരികളും n നിരകളും ഉള്ള ഒരു മെട്രിക്സ് നിർമിക്കുന്നു

```
#include <iostream>
using namespace std;
int main()
{   int m, n, row, col, mat[10][10];
```

```
    cout<< "Enter the order of matrix: ";
    cin>> m >> n;
    cout<<"Enter the elements of matrix\n";
    for (row=0; row<m; row++)
        for (col=0; col<n; col++)
            cin>>mat[row][col];
    cout<<"The given matrix is:";
    for (row=0; row<m; row++)
    {
        cout<<endl;
        for (col=0; col<n; col++)
            cout<<mat[row][col]<<"\t";
    }
    return 0;
}
```

മെട്രിക്സ് നിർമ്മാണം

അംഗങ്ങളെ മെട്രിക്സ് മാതൃകയിൽ പ്രദർശിപ്പിക്കൽ

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
Enter the order of matrix: 3  4
Enter the elements of matrix
1  2  3  4  2  3  4   5   3  4   5  6
The given matrix is:
1    2    3    4
2    3    4    5
3    4    5    6
```

[redacted]

രണ്ടു മെട്രിക്സുകളുടെ ക്രമവും അംഗങ്ങളേയും സ്വീകരിച്ച് അവയുടെ തുക കണ്ടുപിടിക്കുന്നതിനുള്ള ഒരു പ്രോഗ്രാം നമുക്ക് ചർച്ച ചെയ്യാം.

പ്രോഗ്രാം 8. 8 രണ്ട് മെട്രിക്സുകളുടെ തുക കണ്ടെത്തുന്നതിന്

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{   int m1, n1, m2, n2, row, col;
    int A[10][10], B[10][10], C[10][10];
    cout<<"Enter the order of first matrix: ";
    cin>>m1>>n1;
    cout<<"Enter the order of second matrix: ";
```

exit() ഫംഗ്ഷൻ ഉപയോഗിക്കുന്നതിനായി

```
    cin>>m2>>n2;
    if(m1!=m2 || n1!=n2)
    {
        cout<<"Addition is not possible";
        exit(0);
    }
    cout<<"Enter the elements of first matrix\n";
    for (row=0; row<m1; row++)
        for (col=0; col<n1; col++)
            cin>>A[row][col];
    cout<<"Enter the elements of second matrix\n";
    for (row=0; row<m2; row++)
        for (col=0; col<n2; col++)
            cin>>B[row][col];
    for (row=0; row<m1; row++)
        for (col=0; col<n1; col++)
            C[row][col] = A[row][col] + B[row][col];
    cout<<"Sum of the matrices:\n";
    for(row=0; row<m1; row++)
    {
        cout<<endl;
        for (col=0; col<n1; col++)
            cout<<C[row][col]<<"\t";
    }
}
```

2 മെട്രിക്സുകളുടെ ഓർഡർ വ്യത്യാസമാണെങ്കിൽ പ്രോഗ്രാം അവസാനിപ്പിക്കുന്നു.

ആദ്യ മെട്രിക്സിന്റെ നിർമ്മാണം

രണ്ടാം മെട്രിക്സിന്റെ നിർമ്മാണം

മെട്രിക്സിന്റെ തുക കണ്ടുപിടിക്കുന്നു

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
Enter the order of first matrix: 3   4
Enter the order of second matrix: 3   4
Enter the elements of first matrix
2     5     -3    7
5     12    4     9
-3    0     6     -5
Enter the elements of second matrix
1     4     3     5
4     -5    7     13
3     -4    7     9
Sum of the matrices:
3     9     0     12
9     7     11    22
0     -4    13    4
```

ഇവിടെ അംഗങ്ങളെ നൽകിയിരിക്കുന്നത് മെട്രിക്സ് മാതൃകയിലാണ്, പക്ഷെ ഇങ്ങനെ നൽകൽ നിർബന്ധമില്ല

പ്രോഗ്രാം 8.8 ൽ C[i][j] = A[i][j] + B[i][j] എന്നതിനു പകരം C[i][j] = A[i][j] – B[i][j] ഉപയോഗിച്ചാൽ രണ്ട് മെട്രിക്സുകൾ തമ്മിലുള്ള വ്യത്യാസം കണ്ടുപിടിക്കാൻ കഴിയും.

ഇനി നമുക്ക് സമചതുര മെട്രിക്സിന്റെ വികർണ്ണ അംഗങ്ങളുടെ തുക കണ്ടുപിടിക്കാൻ ഒരു പ്രോഗ്രാം എഴുതാം. ഒരു മെട്രിക്സിലെ വരികളുടേയും നിരകളുടേയും എണ്ണം ഒരേ പോലെയാണെങ്കിൽ അത്തരം മെട്രിക്സ് ഒരു സമചതുര മെട്രിക്സ് ആയിരിക്കും. ഒരു സമചതുര മെട്രിക്സിന് രണ്ടു വികർണ്ണങ്ങൾ ഉണ്ട്. `mat[0][0]`, `mat[1][1]`, `mat[2][2]`, . . ., `mat[n-1][n-1]`, അംഗങ്ങളെ മുൻനിര അല്ലെങ്കിൽ മുഖ്യ വികർണ്ണ അംഗങ്ങൾ എന്ന് വിളിക്കുന്നു. മുഖ്യ വികർണ അംഗങ്ങളുടെ തുക കണ്ടുപിടിക്കുന്നതിന് പ്രോഗ്രാം 8.9 ഉപയോഗിക്കാം.

പ്രോഗ്രാം 8.9: ഒരു മെട്രിക്സിന്റെ മുഖ്യ വികർണ്ണ അംഗങ്ങളുടെ തുക കണ്ടെത്തുക.

```
#include <iostream>
using namespace std;
int main()
{   int mat[10][10], n, i, j, s=0;
    cout<<"Enter the rows/columns of square matrix: ";
    cin>>n;
    cout<<"Enter the elements\n";
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            cin>>mat[i][j];
    cout<<"Major diagonal elements are\n";
    for(i=0; i<n; i++)
    {
        cout<<mat[i][i]<<"\t";
        s = s + mat[i][i];
    }
    cout<<"\nSum of major diagonal elements is: ";
    cout<<s;
    return 0;
}
```

തുക കാണുന്നതിന് വികിർണ അംഗങ്ങളെ മാത്രം ഉപയോഗിക്കുന്നു.

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
Enter the rows/columns of square matrix: 3
Enter the elements
3    5    -2
7    4    0
2    8    -1
Major diagonal elements are
3    4    -1
Sum of major diagonal elements is: 6
```

ഓരോ മെട്രിക്സിനും ഒരു ട്രാൻസ്പോസ് ഉണ്ട്. വരിയിലെ അംഗങ്ങൾ നിരയായും അല്ലെങ്കിൽ തിരിച്ചും മാറ്റം വരുത്തിയാണ് ഇത് ലഭിക്കുന്നത്. പ്രോഗ്രാം 8.10 ഈ പ്രക്രിയ വ്യക്തമാക്കുന്നു.

പ്രോഗ്രാം 8.10: ഒരു മെട്രിക്സിന്റെ ട്രാൻസ്പോസ് കണ്ടുപിടിക്കുക.

```
#include <iostream>
using namespace std;
int main()
{   int ar[10][10], m, n, row, col;
    cout<<"Enter the order of matrix: ";
    cin>>m>>n;
    cout<<"Enter the elements\n";
    for(row=0; row<m; row++)
        for(col=0; col<n; col++)
            cin>>ar[row][col];
    cout<<"Original matrix is\n";
    for(row=0; row<m; row++)
    {
        cout<<"\n";
        for(col=0; col<n; col++)
            cout<<ar[row][col]<<"\t";
    }
    cout<<"\nTranspose of the entered matrix is\n";
    for(row=0; row<n; row++)
    {
        cout<<"\n";
        for(col=0; col<m; col++)
            cout<<ar[col][row]<<"\t";
    }
    return 0;
}
```

വരികളുടെയും നിരകളുടെയും എണ്ണം പരസ്പരം മാറ്റിയത് ശ്രദ്ധിക്കുക

സൂചികകളും പരസ്പരം മാറിയത് ശ്രദ്ധിക്കുക.

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
Enter the order of matrix: 4   3
Enter the elements
3    5    -1
2    12   0
6    8    4
7    -5   6
Original matrix is
3    5    -1
```

ഈ അംഗങ്ങളെ ഒരു വരിയിലായും നൽകാവുന്നതാണ്.

```
2     12    0
6     8     4
7     -5    6
Transpose of the entered matrix is
3     2     6     7
5     12    8     -5
-1    0     4     6
```

ഡാറ്റ പട്ടിക രൂപത്തിൽ ക്രമീകരിക്കപ്പെടുമ്പോൾ, ചില സാഹചര്യങ്ങളിൽ നമുക്ക് ഓരോ വരിയിലേയും, നിരയിലേയും അംഗങ്ങളുടെ ആകെത്തുക ആവശ്യമായി വരും. പ്രോഗ്രാം 8.11 ഈ ചുമതല നിർവ്വഹിക്കാൻ കമ്പ്യൂട്ടറിനെ സഹായിക്കുന്നു.

പ്രോഗ്രാം 8.11

```
#include <iostream>
using namespace std;
int main()
{
int ar[10][10], rsum[10]={0}, csum[10]={0};
int m, n, row, col;
cout<<"Enter the number of rows & columns in the array: ";
cin>>m>>n;
cout<<"Enter the elements\n";
for(row=0; row<m; row++)
    for(col=0; col<n; col++)
        cin>>ar[row][col];
for(row=0; row<m; row++)
    for(col=0; col<n; col++)
    {
        rsum[row] += ar[row][col];
        csum[col] += ar[row][col];
    }
cout<<"Row sum of the 2D array is\n";
for(row=0; row<m; row++)
    cout<<rsum[row]<<"\t";
cout<<"\nColumn sum of the 2D array is\n";
for(col=0; col<n; col++)
    cout<<csum[col]<<"\t";
return 0;
}
```

വരിയിലെയും നിരയിലെയും അംഗങ്ങളെ പ്രത്യേകം കൂട്ടു കയും, ഓരോ തുകയും പ്രസ്തുത അറെയിലെ അനുസ്യത മായ സ്ഥാനങ്ങളിൽ സൂക്ഷിക്കു കയും ചെയ്യുന്നു.

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
Enter the number of rows & columns in the array: 3   4
Enter the elements
3     12    5     0
4     -6    2     1
5     7     -6    2
Row sum of the 2D array is
20    1     8
Column sum of 2D array is
12    13    1     3
```

8.4 ബഹുമുഖ അറേകൾ (Multi dimensional arrays)

ഒരു ദ്വിമാന അറെയുടെ ഓരോ അംഗവും മറ്റൊരു അറെയായിരിക്കാം. അത്തരത്തിലുള്ള ഒരു അറെയെ 3D (ത്രിമാന അറെ) അറെ എന്ന് പറയുന്നു.

```
data_type  array_name[size_1][size_2][size_3];
```

എന്ന പ്രസ്താവന ഉപയോഗിച്ച് ഒരു ത്രിമാന അറെയുടെ പ്രഖ്യാപനം നടത്താം. മൂന്നു സൂചിക ഉപയോഗിച്ച് ഒരു 3D അറെയുടെ അംഗങ്ങളെ ഉപയോഗിക്കുകകയും ചെയ്യാം. Ar[10][5][3] ഒരു 3D അറെ ആണെങ്കിൽ ആദ്യത്തെ അംഗം Ar [0][0][0] അവസാന അംഗം Ar [9][4][2] ആയിരിക്കും. ഈ അറെയിൽ 150 (10×5×3) അംഗങ്ങൾ അടങ്ങിയിരിക്കാം.

നമുക്ക് സംഗ്രഹിക്കാം

അറെ എന്നാൽ തുടർച്ചയായ മെമ്മറി സ്ഥാനങ്ങളിൽ ശേഖരിച്ചു വച്ചിട്ടുള്ള ഒരേ തരത്തിലുള്ള ഡാറ്റകളുടെ സമൂഹമാണ്. ഒരു പേരിൽ ഒരേ തരത്തിലുള്ള ഒരു കൂട്ടം വിലകൾ ശേഖരിക്കുന്നതിനായി അറെകൾ ഉപയോഗിക്കുന്നു. ഒരു അറെയിലെ എല്ലാ അംഗങ്ങളേയും സൂചികയുടെ സഹായത്താൽ ഉപയോഗിക്കുവാൻ കഴിയും. for ലൂപ്പിന്റെ സഹായത്തോടെ അറെയിലെ അംഗങ്ങളെ എളുപ്പത്തിൽ ഉപയോഗിക്കുന്നു. കടന്നുപോകൽ, ക്രമപ്പെടുത്തൽ, തിരയൽ തുടങ്ങിയ പ്രവർത്തനങ്ങൾ അറെകളിൽ നടത്തപ്പെടുന്നു. അറെയിലെ അംഗങ്ങളെ ക്രമീകരിക്കുന്നതിന് ബബിൾ സോർട്ട്, സെലക്ഷൻ സോർട്ട് എന്നീ രീതികൾ ഉപയോഗിക്കുന്നു. ഒരു അറെയിൽ ഒരു അംഗത്തെ തിരയാൻ രേഖീയ തിരയൽ, ബൈനറി തിരയൽ എന്നീ വിദ്യകൾ ഉപയോഗിക്കുന്നു. മെട്രിക്സ് സംബന്ധമായ കാര്യങ്ങൾ ചെയ്യുന്നതിന് ദ്വിമാന അറെകൾ ഉപയോഗിക്കുന്നു. ദ്വിമാന അറെയിലുള്ള ഒരു അംഗത്തെ പരാമർശിക്കുന്നതിന് നമുക്ക് രണ്ട് സൂചികകൾ ഉണ്ടാകും. ദ്വിമാന അറെകൾ കൂടാതെ, ബഹുമുഖ അറെകളും സൃഷ്ടിക്കാൻ കഴിയും.

പഠനനേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ പൂർത്തീകരണത്തിനുശേഷം പഠിതാവിന് താഴെ പറയുന്നവ ആർജിക്കാൻ കഴിയും.

- അറെ ഉപയോഗിക്കേണ്ട സാഹചര്യങ്ങളുടെ തിരിച്ചറിവ്
- ഏകമാന, ദ്വിമാന അറെകളുടെ പ്രഖ്യാപനവും പ്രാരംഭവില നൽകലും
- തിരയൽ, ക്രമപ്പെടുത്തൽ തുടങ്ങി വിവിധങ്ങളായ അറെ പ്രവർത്തനങ്ങളുടെ യുക്തി നിർമ്മാണം
- ദ്വിമാന അറെയുടെ സഹായത്തോടെ മെട്രിക്സുമായി ബന്ധപ്പെട്ട പ്രശ്ന പരിഹാരങ്ങൾ

ലാബ് പ്രവർത്തനങ്ങൾ

1. 12 മാസത്തെ വിൽപ്പനയുടെ തുക SalesAmt എന്ന അറെയിലേക്ക് ഇൻപുട്ട് ചെയ്തതിനുശേഷം വിൽപനയുടെ ആകെത്തുകയും ശരാശരിയും കണ്ടെത്തുന്നതിനുള്ള ഒരു C++ പ്രോഗ്രാം എഴുതുക.
2. N സംഖ്യകളുടെ ഒരു അറെ നിർമ്മിച്ചതിന് ശേഷം സംഖ്യകളുടെ ശരാശരി കണ്ടെത്തുകയും ശരാശരിക്ക് മുകളിൽ ഉള്ള സംഖ്യകൾ പ്രദർശിപ്പിക്കുകയും ചെയ്യുന്നതിന് ഒരു C++ പ്രോഗ്രാം എഴുതുക.
3. വില, അളവ്, തൂക്കം എന്നിവ ശേഖരിക്കുന്നതിനായി price, quanttiy, amount എന്നിങ്ങനെ 3 അറെകൾ നിർമ്മിക്കുക. ഓരോ അറെയും 10 അംഗങ്ങളെ ഉൾക്കൊള്ളാൻ കഴിയുന്നവയായിരിക്കണം. price, quanttiy എന്നീ അറകളിലേക്ക് വിലകൾ നൽകുക. amount അറെയുടെ മൂല്യം amount[i] = price[i] × quanttiy[i] എന്നായിരിക്കണം. എല്ലാ ഡാറ്റയും നൽകിയ ശേഷം, താഴെ കൊടുത്തിരിക്കുന്ന രീതിയിൽ ഔട്ട്പുട്ട് പ്രദർശിപ്പിക്കുന്നതിനു വേണ്ടിയുള്ള ഒരു C++ പ്രോഗ്രാം എഴുതുക

Price	Quantity	Amount
______	______	______
______	______	______

4. ഒരു അറെയിലേക്ക് 10 സംഖ്യകൾ നൽകിയതിന് ശേഷം, അവയിലെ ഏറ്റവും വലിയ സംഖ്യയും ചെറിയ സംഖ്യയും കണ്ടുപിടിക്കുന്നതിനുള്ള ഒരു C++ പ്രോഗ്രാം എഴുതുക.
5. ഓർഡർ n ആയിട്ടുള്ള ഒരു സമചതുര മെട്രിക്സിന്റെ വികർണ്ണത്തിനു മുകളിലുള്ള അംഗങ്ങളെ പ്രദർശിപ്പിക്കുന്നതിനുള്ള C++ പ്രോഗ്രാം എഴുതുക. ഉദാഹരണത്തിന് താഴെ കാണുന്ന മെട്രിക്സ് പരിഗണിക്കുക.

2	3	1
7	1	5
2	5	1

ഉത്തരം ഇവിടെ കാണുന്ന വിധമായിരിക്കും

2	3	1
	1	5
		1

5. ഓർഡർ n ആയിട്ടുള്ള ഒരു സമചതുര മെട്രിക്സിന്റെ വികർണ്ണത്തിനു താഴെയുള്ള അംഗങ്ങളെ പ്രദർശിപ്പിക്കുന്നതിനുള്ള C++ പ്രോഗ്രാം എഴുതുക. ഉദാഹരണത്തിന് താഴെ കാണുന്ന മെട്രിക്സ് പരിഗണിക്കുക.

2	3	1
7	1	5
2	5	1

ഉത്തരം ഇവിടെ കാണുന്ന വിധമായിരിക്കും

2		
7	1	
2	5	7

7. താഴെ കാണിച്ചിരിക്കുന്നതുപോലെ പാസ്കൽസ് ത്രികോണം (Pascal's Triangle) പ്രദർശിപ്പിക്കുന്നതിനുള്ള ഒരു C++ പ്രോഗ്രാം എഴുതുക

1				
1	2	1		
1	3	3	1	
1	4	6	4	1

മാതൃകാ ചോദ്യങ്ങൾ

1. ഒരു അറെയിലെ എല്ലാ അംഗങ്ങളും ________ ഡാറ്റ ഇനം ആയിരിക്കണം.
2. പത്തു അംഗങ്ങളുള്ള ഒരു അറെയുടെ സൂചിക ________ മുതൽ ________ വരെയുള്ള സംഖ്യകൾ ആയിരിക്കും.
3. ഒരു അറെയിലെ അംഗത്തെ ________ ഉപയോഗിച്ച് ഉപയോഗിക്കാം.
4. AR ഒരു അറെയാണെങ്കിൽ, AR[7] എത് അംഗത്തെ പ്രതിനിധാനം ചെയ്യുന്നു?
5. int a[3]={2,3,4}; എന്ന അറെയിൽ a[1] ന്റെ വില എന്ത്?
6. int a[]={1,2,3,4}; എന്ന അറെയിൽ a[2] ന്റെ വില എന്ത്?
7. int a[5]={1,2,3,4}; എന്ന അറെയിൽ a[4] ന്റെ വില എന്ത്?
8. 89, 75, 82, 93, 78, 95. എന്നീ സ്കോറുകൾ score എന്ന അറെയിലേക്ക് പ്രാരംഭ വിലയായി നല്കുന്നതിനുള്ള പ്രസ്താവന എഴുതുക
9. ഒരു അറെയിലെ എല്ലാ അംഗങ്ങളേയും പ്രദർശിപ്പിക്കുന്നത് ________ പ്രവർത്തനത്തിന് ഒരു ഉദാഹരണമാണ്.

10. int a[2][3]; എന്ന അറെ നിർമിക്കുന്നതിന് എത്ര ബൈറ്റുകൾ ആവശ്യമാണ്.
11. m അംഗങ്ങളുള്ള അറെയിൽ, ബൈനറി തിരയലിൽ ഒരു അംഗത്തെ കണ്ടെത്തുന്നതിന് പരമാവധി n തിരയൽ ആവശ്യമാണ്. എന്നാൽ അംഗങ്ങളുടെ എണ്ണം ഇരട്ടിയെങ്കിൽ എത്ര തിരയൽ ആവശ്യമായി വരും?
12. Mark എന്ന അറെയിലേക്ക് പൂജ്യം പ്രാരംഭ വിലയായി നല്കുന്നതിനുള്ള പ്രസ്താവന എഴുതുക.
13. പ്രസ്താവന ശരിയോ തെറ്റോ: 10 അംഗങ്ങളുള്ള അറെയിലെ പതിനാറാമത്തെ അംഗത്തെ ഉപയോഗിക്കുവാൻ നിങ്ങൾ ശ്രമിക്കുകയാണെങ്കിൽ കമ്പൈലർ തെറ്റ് രേഖപ്പെടുത്തും.

ലഘു ഉപന്യാസ ചോദ്യങ്ങൾ

1. അറെ നിർവ്വചിക്കുക.
2. int studlist[1000]; എന്ന പ്രഖ്യാപന പ്രസ്താവന അർത്ഥമാക്കുന്നത് എന്ത്?
3. ഒരു ഏകമാന അറെയ്ക്കായി മെമ്മറി അനുവദിക്കുന്നത് എങ്ങനെ?
4. 10 അംഗങ്ങളുള്ള അറെയിലേക്ക് സംഖ്യകളെ സ്വീകരിച്ച് അവയിലെ ഒറ്റ സംഖ്യയുടെയും ഇരട്ട സംഖ്യയുടെയും എണ്ണം പ്രദർശിപ്പിക്കുന്നതിനുള്ള C++ കോഡ് ശകലങ്ങൾ എഴുതുക.
5. 2, 3, 4, 5 എന്നീ സംഖ്യകൾ ഒരു അറെയിൽ ശേഖരിക്കുന്നതിനുള്ള പ്രാരംഭ വിലനൽകൽ പ്രസ്താവന എഴുതുക.
6. കടന്നുപോകൽ എന്നാൽ എന്ത്?
7. ക്രമപ്പെടുത്തൽ നിർവ്വചിക്കുക.
8. തിരയൽഎന്നാൽ എന്ത്?
9. ബബിൾ സോർട്ട് എന്നാൽ എന്ത്?
10. ബൈനറി തിരയൽ എന്നാൽ എന്ത്?
11. ഒരു ദ്വിമാന അറെ നിർവ്വചിക്കുക
12. ഒരു ദ്വിമാന അറെയ്ക്കായി മെമ്മറി അനുവദിക്കുന്നത് എങ്ങനെ?

ഉപന്യാസ ചോദ്യങ്ങൾ

1. അറെ AR 25, 81, 36, 15, 45, 58, 70 എന്നീ അംഗങ്ങൾ ഉൾക്കൊള്ളുന്നു. 45 എന്ന വില തിരയുന്നതിനായുള്ള ബൈനറി തിരയൽ രീതിയുടെ പ്രവർത്തനം വിശദമാക്കുക.
2. തുല്യ വലിപ്പത്തിലുള്ള രണ്ടു അറെയിൽ അംഗങ്ങളെ സ്വീകരിച്ച് അതത് സ്ഥാനങ്ങളിലെ അംഗങ്ങൾ തമ്മിലുള്ള വ്യത്യാസം കണ്ടെത്തുന്നതിനുള്ള C++ പ്രോഗ്രാം എഴുതുക.

3. 3, 32, 25, 44, 16, 37, 12 എന്നീ അംഗങ്ങളെ ക്രമീകരിക്കുന്നതിനുള്ള ബബിൾ സോർട്ടിന്റെ പ്രവർത്തനരീതി വിശദീകരിക്കുക.

4. 24, 45, 98, 56, 76, 24, 15 എന്നീ അംഗങ്ങളെ ക്രമീകരിക്കുന്നതിനുള്ള രേഖീയ തിരയലിന്റെ പ്രവർത്തനരീതി വിശദീകരിക്കുക.

5. രണ്ട് മെട്രിക്സുകൾ തമ്മിൽ വ്യവകലനം ചെയ്യുന്നതിനുള്ള ഒരു പ്രോഗ്രാം എഴുതുക.

6. ഒരു ദ്വിമാന അറെയിൽ അംഗങ്ങളുടെ തുകയും ശരാശരിയും കണ്ടെത്താനായി പ്രോഗ്രാം എഴുതുക.

7. ഒരു ദ്വിമാന അറെയിലെ ഏറ്റവും വലിയ സംഖ്യ കണ്ടെത്തുന്നതിനുള്ള പ്രോഗ്രാം എഴുതുക.

****