2 അറെകൾ

പ്രധാന പഠനനേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ പഠനം പൂർത്തിയാക്കുന്നതോടെ പഠിതാവ് ആർജിക്കേണ്ട പഠനനേട്ടങ്ങൾ:

- അറെ ഉപയോഗിക്കേണ്ട വ്യത്യസ്ത സാഹചര്യങ്ങൾ തിരിച്ചറിയുന്നു.
- ഡാറ്റ സമൂഹത്തെ സൂചിപ്പിക്കുന്നതിനായി അറെ ഉപയോഗിക്കുന്നു.
- അറെയുടെ മെമ്മറി നീക്കി വയ്ക്കൽ മനസിലാക്കുന്നു.
- പ്രശ്ന നിർധാരണത്തിനായി അറെ അംഗങ്ങളെ ഉപയോഗിക്കുന്നു.
- സ്ട്രിങ്ങുകൾ പ്രതിനിധികരിക്കുന്നതിനായി ക്യാരക്റ്റർ അറെ ഉപയോഗിക്കുന്നു.
- വ്യത്യസ്ത വേഡ് പ്രോസസ്സിങ് പ്രവർത്തനങ്ങൾക്കായി ക്യാരക്ടർ അറെ ഉപയോഗിക്കുന്നു.

I5D2Z5

പ്രോഗ്രാമുകളിൽ ഡാറ്റ സംഭരിക്കുന്നതിനായി നാം വേരിയബളുകൾ ഉപയോഗിക്കുന്നു. എന്നാൽ ഡാറ്റയുടെ എണ്ണം കൂടുതലാണെങ്കിൽ കൂടുതൽ വേരിയബളുകൾ 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.

2.1 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 അറെ എന്ന ആശയം നമുക്ക് ഉപകരിക്കും. അറെയിലെ ഓരോ അംഗങ്ങൾക്കും മെമ്മറി സ്ഥാനങ്ങൾ അനുവദിക്കേണ്ടതുണ്ട്. മെമ്മറി നീക്കിവെയ്ക്കുന്നതിന് പ്രഖ്യാപന പ്രസ്താവനകൾ ആവശ്യമാണെന്നും നമുക്കറിയാം. എങ്ങനെയാണ് അറെകൾ പ്രഖ്യാപനം നടത്തി അവ ഉപയോഗിക്കുന്നത് എന്ന് നമുക്ക് നോക്കാം.

2.1.1 അറെകളുടെ പ്രഖ്യാപനം (Array Declaration)

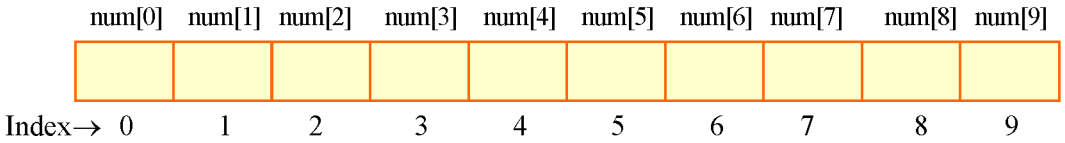
സാധാരണ വേരിയബിളിനെ പോലെ അറെ ഉപയോഗിക്കുന്നതിന് മുമ്പായി പ്രഖ്യാപനം നടത്തേണ്ടതുണ്ട്. C++ൽ അറെ പ്രഖ്യാപനം ചെയ്യുന്നതിനുള്ള വാക്യഘടന താഴെ പറഞ്ഞിരിക്കുന്നു.

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

വാക്യഘടനയിൽ `datatype` എന്നത് അറെയിലെ അംഗങ്ങളുടെ ഡേറ്റയുടെ ഇനമാണ് സൂചിപ്പിക്കുന്നത്. `array_name` എന്നത് അറെയുടെ പേരും `size` എന്നത് അറെയിലെ ആകെ അംഗങ്ങളുടെ എണ്ണം വ്യക്തമാക്കുന്ന ഒരു പോസിറ്റീവ് സംഖ്യയും ആകുന്നു. താഴെ പറയുന്നത് ഒരു അറെ നിർമാണത്തിന്റെ ഉദാഹരണമാണ്.

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

മുകളിലുള്ള പ്രസ്താവന `num` എന്ന് വിളിക്കുന്ന 10 പൂർണ്ണസംഖ്യകൾ സൂക്ഷിക്കാവുന്ന ഒരു അറെയെ നിർമിക്കുന്നു. ചിത്രം 8.1 കാണിച്ചിരിക്കുന്നതു പോലെ അറെയിലെ അംഗങ്ങൾ മെമ്മറിയിൽ തുടർച്ചയായി സൂക്ഷിക്കുന്നു.



ചിത്രം 2.1 ഒരു അറെയിലെ അംഗങ്ങളുടെ ക്രമീകരണം

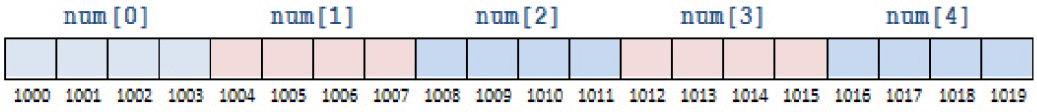
അറെയിലെ അംഗങ്ങൾ ക്രമാനുഗതമായി സൂക്ഷിക്കുന്നതുകൊണ്ട്, ഏത് അംഗത്തേയും അറെയുടെ പേരും അംഗത്തിന്റെ സ്ഥാനവും നൽകി ഉപയോഗിക്കുവാൻ കഴിയും. ഓരോ അംഗത്തെയും സൂചിപ്പിക്കുന്ന സ്ഥാനത്തിന് സൂചിക (index or subscript) എന്നു പറയുന്നു. C++ൽ അറെയുടെ സൂചിക പൂജ്യത്തിൽ ആരംഭിക്കുന്നു. `int num[10]` എന്ന് ഒരു അറെ നിർമ്മിച്ചാൽ 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 വിദ്യാർഥികളുടെ 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.

2.1.2 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 ആയതിനാൽ, ഓരോ അംഗത്തിന്റെയും വ്യാപ്തി 4 ബൈറ്റുകൾ ആണ് (16 ബിറ്റ് പ്രതിനിധീകരിക്കുന്ന 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 2.2 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 × 10 = 40 ബൈറ്റുകൾ ആയിരിക്കും.

2.1.3 അറെയുടെ പ്രാരംഭ വില നൽകൽ (Array Initialization)

സാധാരണ വേരിയബിൾ പോലെ തന്നെ അറെയുടെ പ്രഖ്യാപന പ്രസ്താവനകളോടൊപ്പം അവയുടെ പ്രാരംഭ വിലകൾ നൽകുവാൻ കഴിയും. താഴെപ്പറയുന്ന ഉദാഹരണങ്ങളിൽ കാണിച്ചിരിക്കുന്നതുപോലെ അറെയിലെ അംഗങ്ങളെ ബ്രാക്കറ്റിനുള്ളിൽ എഴുതണം.

```
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};
```

അറെയിലെ അംഗങ്ങളെ അവ എഴുതപ്പെട്ട ക്രമത്തിൽ സൂക്ഷിക്കുന്നു. ഒന്നാമത്തെ അംഗം സൂചിക 0 ലും, രണ്ടാമത്തെ അംഗം സൂചിക 1 ലും 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 അറെയിലെ 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};
```

2.1.4 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 2.1: 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:

കുറഞ്ഞത് ഒരിക്കലെങ്കിലും അറെയിലെ ഓരോ അംഗത്തേയും ഉപയോഗിക്കുക എന്നതാണ് കടന്നുപോകൽ എന്നതുകൊണ്ട് ഉദ്ദേശിക്കുന്നത്. അറെയിലെ എല്ലാ അംഗങ്ങളും പ്രദർശിപ്പിക്കുന്നത് കടന്നുപോകൽ ഒരു ഉദാഹരണമാണ്. ഏതെങ്കിലുമൊരു പ്രവർത്തനം അറെയിലെ എല്ലാ അംഗങ്ങളിലും നടക്കുന്നു എങ്കിൽ അതിനെ കടന്നുപോകൽ എന്നു പറയുന്നു. ഒരു പ്രോഗ്രാമിൽ എങ്ങനെയാണ് കടന്നുപോകൽ നടത്തുന്നത് എന്നത് പ്രോഗ്രാം 2.2 ൽ കൊടുത്തിരിക്കുന്നു

പ്രോഗ്രാം 2.2: അറെയിലെ കടന്നുപോകൽ

```
#include <iostream>
using namespace std;
int main()
{
int a[5], i;
cout<<"Enter the elements of the array :";
for(i=0; i<5; i++)
     cin >> a[i];
for(i=0; i<5; i++)
     a[i] = a[i] + 1;
cout<<"\nNow value of the elements in the array are...\n";
for(i=0; i<5; i++)
    cout<< a[i]<< "\t";
return 0;
}
```

കടന്നുപോകൽ

കടന്നുപോകൽ

കടന്നുപോകൽ

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
അറെ അംഗങ്ങളെ ചേർക്കുക: 12   3    6    1    8
ഇപ്പോൾ അറെയിലെ ഓരോ അംഗത്തിന്റെയും വില: 1    3    6    8    12
```

പ്രോഗ്രാം 2.3: അറെ അംഗങ്ങളുടെ തുക കണ്ടെത്തുക

```
#include <iostream>
using namespace std;
int main()
{   int a[5], i, sum;
    cout<<"Enter the elements of the array:";
    for (i=0; i<5; i++)
         cin>>a[i]; // Reading the elements
    sum = 0
    for (i=0; i<5; i++)
         sum = sum + a[i]; // a case of traversal
    cout<<"\nsum of the elements of the array is "<<sum;
    return 0;
}
```

ഔട്ട്പുട്ടിന്റെ മാതൃക:

```
അറെ അംഗങ്ങളെ ചേർക്കുക: 12   3    6    1    8
അറെ അംഗങ്ങളുടെ തുക: 30
```

പ്രോഗ്രാം 2.4: അറെയിലെ ഏറ്റവും വലിയ അംഗത്തെ കണ്ടെത്തുക

```
#include <iostream>
using namespace std;
int main()
{
    int a[5],i, big;
    cout<<"Enter the elements of the array:";
    for(i=0; i<5; i++)
        cin>>a[i];
    big = a[0]
    for(i=1; i<5; i++)
            if(a[i] > big)// a case of traversal
                big = a [i]
    cout<<"nThe biggest element is "<< big;
    return 0;
}
```

ഔട്ട്പുട്ടിന്റെ മാതൃക:

അറെ അംഗങ്ങളെ ചേർക്കുക: 23 10 -3 7 11

അറെയിലെ ഏറ്റവും വലിയ അംഗം: 12

2.2. അറെ ഉപയോഗിച്ചുള്ള സ്ട്രിങ് കൈകാര്യം ചെയ്യൽ (String handling using arrays)

C++ ലെ ഒരുതരം ലിറ്ററലാണ് സ്ട്രിങ്. പ്രോഗ്രാമുകളിൽ ഇവ കാണപ്പെടുന്നത് ഉദ്ധരണിക്കുള്ളിൽ (Double quotes) തുടർച്ചയായുള്ള കാരക്റ്ററുകളായാണ്. നിങ്ങളോട് പേര് ശേഖരിക്കുവാനും പ്രദർശിപ്പിക്കുന്നതിനുമുള്ള ഒരു പ്രോഗ്രാം എഴുതുവാൻ ആവശ്യപ്പെട്ടുവെന്നിരിക്കട്ടെ. ഡാറ്റ ശേഖരിക്കുവാൻ വേരിയബിൾ ആവശ്യമാണെന്ന് ഇതിനു മുമ്പ് നാം പഠിച്ചിട്ടുണ്ട്. my_name എന്ന വേരിയബിൾ ഒരു ഐഡന്റിഫയർ ആയി ഇവിടെ നമുക്ക് ഉപയോഗിക്കാം. ഒരു വേരിയബിൾ ഉപയോഗിക്കുന്നതിനു മുമ്പ് അത് പ്രഖ്യാപിക്കണമെന്നുള്ളത് ഈ അവസരത്തിൽ തീർച്ചയായും ഓർമിക്കേണ്ടതാണ്. സ്ട്രിങ് ഡാറ്റയെ സൂചിപ്പിക്കാനുള്ള അടിസ്ഥാന ഡാറ്റ ഇനം നിലവിലില്ലാത്തതിനാൽ ഏതു തരം ഡാറ്റയാണ് സ്ട്രിങ് ഡാറ്റ ശേഖരിക്കുന്ന വേരിയബിൾ പ്രഖ്യാപനത്തിന് ഉപയോഗിക്കാനാവുക എന്ന് പറയാൻ സാധിക്കില്ല? അതുകൊണ്ട് നമുക്ക് char ഡാറ്റ ഇനത്തെക്കുറിച്ച് ആലോചിക്കാം. എന്നാൽ അവിടെയും ഒരു പ്രശ്നമുണ്ട്. char ഡാറ്റ ഇനത്തിന് ഒരു കാരക്റ്റർ മാത്രമേ ശേഖരിക്കുവാൻ കഴിയുകയുള്ളൂ. അതുകൊണ്ടുതന്നെയാണ് സ്ട്രിങ് എന്നത് തുടർച്ചയായ കാരക്റ്ററുകളുടെ ഇൻപുട്ട് ആയി സ്വീകരിക്കേണ്ടി വരുന്നത് .

"Niketh" എന്ന പേര് പരിഗണിക്കുക. ഇത് ആറ് കാരക്റ്ററുകൾ ഉൾക്കൊള്ളുന്ന ഒരു സ്ട്രിങ് ആണ്. എന്നാൽ ഒരു കാരക്റ്റർ അറെയ്ക്ക് ഒന്നിലധികം കാരക്റ്ററുകളെ ശേഖരിക്കുവാൻ കഴിയുമെന്ന് നമുക്കറിയാം. അതുകൊണ്ടു ഒരു അറേയെ താഴെ കാണുന്നവിധം പ്രഖ്യാപിക്കാവുന്നതാണ്.

```
char my_name[10];
```

my_name എന്ന് പേരുള്ള അറേയിൽ ഒരു ബൈറ്റ് വീതം വലിപ്പമുള്ള തുടർച്ചയായ പത്ത് മെമ്മറി സ്ഥാനങ്ങൾ നീക്കിവച്ചിട്ടുണ്ട്. ഈ അറേയിലേക്ക് താഴെ കാണുന്നത് പോലെ പ്രാരംഭ വിലകൾ നൽകാവുന്നതാണ്.

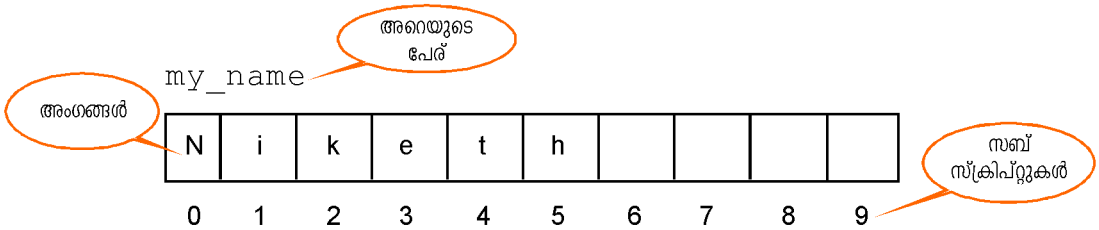
```
char my_name[10] = { 'N','i','k','e','t','h'};
```

ചിത്രം 2.1ൽ മേൽ സൂചിപ്പിച്ച കാരക്റ്റർ അറേയുടെ മെമ്മറി നീക്കിവെയ്പ് ചിത്രീകരിച്ചിട്ടുണ്ട്. സ്ട്രിങ്ങിലെ കാരക്റ്ററുകൾ കോമായുപയോഗിച്ച് വേർതിരിച്ചാണ് ശേഖരിക്കുന്നത് എന്ന് പ്രത്യേകം ശ്രദ്ധിക്കേണ്ടതാണ്. ഇതേ ഡാറ്റ ഇൻപുട്ട് ചെയ്യണമെങ്കിൽ താഴെ പറഞ്ഞിരിക്കുന്ന C++ പ്രസ്താവന ഉപയോഗിക്കാം .

```
for   (int i=0;i<6;i++)

cin>>my_name[i];
```

ഈ കോഡ് പ്രവർത്തിക്കുന്ന സമയത്ത് നാം "Niketh" എന്ന സ്ട്രിങ്ങിനകത്തെ ആറ് കാരക്റ്ററുകൾ ഒന്നിന് പുറകെ ഒന്നായി സ്പേസ് ബാർ, ടാബ് കീ അല്ലെങ്കിൽ എന്റർ കീ എന്നിവയിൽ ഏതെങ്കിലും ഒന്നുപയോഗിച്ച് വേർതിരിച്ച് വേണം ഇൻപുട്ട് ചെയ്യേണ്ടത്. മേൽ സൂചിപ്പിച്ച രണ്ടു രീതിയിലുമുള്ള മെമ്മറി നീക്കിവയ്ക്കലുകൾ താഴെ തന്നിരിക്കുന്ന വിധത്തിലാണ്.



ചിത്രം 2.3: കാരക്റ്റർ അറെയുടെ മെമ്മറി നീക്കിവെയ്പ്

സ്ട്രിങ്ങുകൾ തുടർച്ചയായുള്ള കാരക്റ്ററുകൾ ആയതിനാൽ കാരക്റ്റർ അറേയെ സ്ട്രിങ്ങുകൾ ശേഖരിക്കുന്നതിന് ഉപയോഗിക്കാവുന്നതാണ്. എന്നിരുന്നാലും ഒരു സ്ട്രിങ് നേരിട്ട് ഇൻപുട്ട് ചെയ്യുന്നതായി നമുക്ക് തോന്നുകയേ ഇല്ല എന്നത് ഒരു വസ്തുതയാണ്. പകരം നാം ഒന്നിന് പുറകെ ഒന്നായി കാരക്റ്ററുകൾ ഇൻപുട്ട് ചെയ്ത് അതിനെ ഒരു സ്ട്രിങ് ആക്കി മാറ്റുകയാണ് ചെയ്യേണ്ടത്.

C++ ൽ കാരക്റ്റർ അറേകൾക്ക് ചില പ്രത്യേക സവിശേഷതകൾ ഉണ്ട്. ഒരിക്കൽ ഒരു കാരക്റ്റർ അറെ പ്രഖ്യാപിച്ചാൽ, അറെയുടെ പേര് സ്ട്രിങ് ഡാറ്റ സൂക്ഷിക്കാനുള്ള സാധാരണ വേരിയബിളായിത്തന്നെ പരിഗണിക്കപ്പെടുന്നു. അതുകൊണ്ടു തന്നെ കാരക്റ്റർ അറേയുടെ പേര് സ്ട്രിങ് വേരിയബിളിന് സമാനമാണ് എന്ന് പറയാം. അതിനാൽ നിങ്ങളുടെ പേര് my_name (അറെയുടെ പേര്) ൽ താഴെ കൊടുത്തിട്ടുള്ള പ്രസ്താവന ഉപയോഗിച്ച് സംഭരിക്കാവുന്നതാണ്.

```
cin>>my_name;
```

മറ്റുള്ള ഡാറ്റ ഇനങ്ങളുടെ കാര്യത്തിൽ മേൽ സൂചിപ്പിക്കപ്പെട്ട പ്രയോഗം തെറ്റാണെന്ന് പ്രത്യേകം ശ്രദ്ധിക്കേണ്ടതാണ്. ഇനി നമുക്ക് ഒരു സ്ട്രിങ് ഇൻപുട്ട് ചെയ്തു പ്രദർശിപ്പിക്കുന്നതിനുള്ള പ്രോഗ്രാം പൂർത്തിയാക്കാം. പ്രോഗ്രാം 2.5 ൽ പറഞ്ഞിരിക്കുന്നത് പോലെ ഇത് ചെയ്യാവുന്നതാണ്.

പ്രോഗ്രാം 2.5 ഒരു സ്ട്രിങ് ഇൻപുട്ട് ചെയ്ത് പ്രദർശിപ്പിക്കുക.

```
#include<iostream>
using namespace std;
int main()
{
    char my_name[10];
    cout << "Enter your name: ";
    cin >> my_name;
    cout << "Hello " << my_name;
}
```

ഈ പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുമ്പോൾ താഴെ കാണുന്നവിധം ഔട്ട്പുട്ട് ലഭിക്കുന്നതാണ്.

```
Enter your name: Niketh
Hello Niketh
```

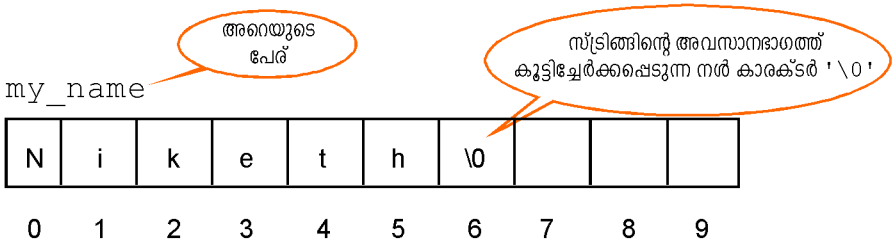
പ്രത്യേകം ശ്രദ്ധിക്കേണ്ടത് ഇവിടെ സ്ട്രിങ് കോൺസ്റ്റന്റ് "Hello" അല്ല "Hello " ആണ് എന്നുള്ളത്. ('o' എന്ന അക്ഷരത്തിനു ശേഷം ഒരു സ്പേസ് നൽകിയിട്ടുണ്ട്).

നമുക്കു ചെയ്യാം

പ്രോഗ്രാം 2.5 പ്രവർത്തിപ്പിച്ച് നിങ്ങളുടെ പേരിന്റെ കൂടെ ഇനിഷ്യലും ഇൻപുട്ട് ചെയ്ത് ഔട്ട്പുട്ട് ശരിയോ തെറ്റോ എന്ന് പരിശോധിക്കുക. പേരിൽ 10 കാരക്റ്ററുകളിലും കൂടുതൽ ഉണ്ടെങ്കിൽ അറെയുടെ വലിപ്പം ആവശ്യത്തിനനുസരിച്ച് വർദ്ധിപ്പിക്കുക .

2.3 സ്ട്രിങ്ങിനു വേണ്ടിയുള്ള മെമ്മറി നീക്കിവെയ്പ് (Memory allocation for strings)

ഒരു അറെയിലുള്ള കാരക്റ്ററുകൾക്ക് എങ്ങനെയാണ് മെമ്മറി അനുവദിക്കുന്നതെന്നു നാം കണ്ടു കഴിഞ്ഞു. ചിത്രം 2.1 ൽ കാണിച്ചിരിക്കുന്നത് പോലെ മെമ്മറി ആവശ്യകത കണക്കാക്കുന്നത് ഇൻപുട്ട് ചെയ്ത കാരക്റ്ററുകളുടെ എണ്ണമനുസരിച്ചാണ്. എന്നാൽ ഒരു കാരക്റ്റർ അറെയിൽ സ്ട്രിങ് ഇൻപുട്ട് ചെയ്യുമ്പോൾ ചിത്രം മറ്റൊന്നാകുന്നു. നമ്മൾ പ്രോഗ്രാം 2.1 പ്രവർത്തിപ്പിച്ച് "Niketh" എന്ന സ്ട്രിങ് ഇൻപുട്ട് ചെയ്താൽ മെമ്മറി നീക്കി വെയ്പ് ചിത്രം 2.5 ൽ താഴെ കാണുന്ന വിധമായിരിക്കും.



ചിത്രം 2.4 : കാരക്റ്റർ അറെയുടെ മെമ്മറി നീക്കിവെയ്പ്.

ഇവിടെ നൾ കാരക്റ്റർ ('\0') സ്ട്രിങ്ങിന്റെ അവസാനഭാഗത്ത് കൂട്ടിച്ചേർക്കപ്പെടുന്നു. ഇത് കാരക്റ്റർ സ്ട്രിങ്ങിന്റെ ടെർമിനേറ്റർ ആയി ഉപയോഗിക്കുന്നു. അതിനാൽ ഒരു സ്ട്രിങ് സംഭരിക്കാനാവശ്യമായ മെമ്മറി എന്നത് സ്ട്രിങ്ങിലെ ആകെ കാരക്റ്ററുകളുടെ എണ്ണവും നൾ കാരക്റ്ററിനു വേണ്ട ഒരു ബൈറ്റും ചേർന്നതാണ്. മേല്പറഞ്ഞ "Niketh" എന്ന സ്ട്രിങ് ശേഖരിക്കുവാൻ ഏഴ് ബൈറ്റ് ആവശ്യമാണ്. (അതായത് 6 കാരക്റ്ററുകൾക്കുള്ള 6 ബൈറ്റ് + നൾ കാരക്റ്ററിനുള്ള 1 ബൈറ്റ്).

താഴെ കാണിച്ചിരിക്കുന്നവിധത്തിൽ നമുക്ക് കാരക്റ്റർ അറെയ്ക്ക് പ്രാരംഭവില നൽകാം.

```
char my_name[10] = "Niketh";
char str[] ="Hello world";
```

ആദ്യത്തെ പ്രസ്താവനയിൽ പത്ത് മെമ്മറി സ്ഥാനങ്ങൾ നീക്കി വെക്കുകയും അതിൽ പ്രാരംഭ വിലയും നൾ കാരക്റ്ററും സംഭരിക്കുകയും ചെയ്യുന്നു. ഇവിടെ അവസാന മൂന്ന് ബൈറ്റുകൾ ഉപയോഗിക്കുന്നില്ല. എന്നാൽ രണ്ടാമത്തെ സ്റ്റേറ്റ്മെന്റിൽ അറെയുടെ വലിപ്പം ഉൾപ്പെടുത്തിയിട്ടില്ല. അതുകൊണ്ട് 11 ബൈറ്റ് സ്ട്രിങ്ങിനും 1 ബൈറ്റ് '\0' നും അടക്കം ആകെ 12 ബൈറ്റ് നീക്കിവെയ്ക്കപ്പെടുന്നു.

2.4 സ്ട്രിങ്ങിനു മേലുള്ള ഇൻപുട്ട്/ഔട്ട്പുട്ട് പ്രവർത്തനങ്ങൾ (Input/Output operations on strings)

പ്രോഗ്രാം 2.5 ൽ സ്ട്രിങ് ഡാറ്റ ഇൻപുട്ട്/ഔട്ട്പുട്ട് ചെയ്യുന്നതിനുള്ള പ്രസ്താവനകൾ ഉൾപെടുത്തിയിട്ടുണ്ട്. അറെയുടെ വലിപ്പം 20 ആക്കി പ്രഖ്യാപന പ്രസ്താവനയിൽ ഒരു ചെറിയ മാറ്റം വരുത്തുക. "Maya Mohan" എന്ന പേര് ഇൻപുട്ട് ചെയ്ത് പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുകയാണെങ്കിൽ താഴെ കാണുന്ന വിധത്തിലുള്ള ഔട്ട്പുട്ട് ലഭിക്കുന്നതാണ്.

```
Enter your name: Maya Mohan
Hello Maya
```

സ്ട്രിങ് ശേഖരിക്കുന്നതിന് ആവശ്യമായ വലിപ്പം അറെയ്ക് ഉണ്ടെങ്കിലും നമുക്ക് ഔട്ട്പുട്ടായി "Maya" എന്ന് മാത്രമാണ് ലഭിക്കുന്നത്. ഇതെന്തുകൊണ്ട് സംഭവിച്ചു?

നമുക്ക് `cin>>my_name;` എന്ന പ്രസ്താവന സൂക്ഷ്മമായൊന്നു പരിശോധിക്കാം. ഒരു ഡാറ്റ ഇനത്തെ മാത്രമേ ഈ പ്രസ്താവന ഉപയോഗിച്ചു ഇൻപുട്ട് ചെയ്യാൻ കഴിയൂ എന്ന് നമുക്കറിയാം. ഒരു ഡാറ്റയെ മറ്റൊന്നിൽ നിന്ന് വേർതിരിക്കുവാൻ ഉപയോഗിക്കുന്നതാണ് വൈറ്റ് സ്പേസ്. അതുകൊണ്ട് "Maya Mohan" എന്നത് രണ്ട് ഡാറ്റയായി പരിഗണിക്കപ്പെടുന്നു. (Maya, Mohan എന്നിവയ്ക്കിടയ്ക്ക് വൈറ്റ് സ്പേസ് ഉള്ളതുകൊണ്ട്). `my_name`ന് മുമ്പ് ഒരു ഇൻപുട്ട് ഓപ്പറേറ്റർ (>>) മാത്രമേയുള്ളു. അതിനാൽ ആദ്യത്തെ ഡാറ്റയായ "Maya" മാത്രം സംഭരിക്കപ്പെടുന്നു. അതിന് ശേഷമുള്ള വൈറ്റ് സ്പേസ് ഡിലിമിറ്റർ ആയി വർത്തിക്കുകയും ചെയ്യുന്നു. അതിനാൽ ഈ പ്രസ്താവന സംവിധാനം ഉപയോഗിച്ച് വൈറ്റ് സ്പേസ് അടങ്ങിയ സ്ട്രിങ്ങുകൾ മുഴുവനായും ഇൻപുട്ട് ചെയ്യുവാൻ കഴിയുകയില്ല. ഇതിനു പരിഹാരമായി `gets()` എന്ന ഫങ്ഷൻ ഉപയോഗിക്കാവുന്നതാണ്. സ്റ്റാൻഡേർഡ് ഇൻപുട്ട് ഉപകരണങ്ങളിൽ (keyboard) നിന്ന് വൈറ്റ് സ്പേസ് അടങ്ങിയ സ്ട്രിങ്ങുകളെ സ്വീകരിക്കുകയും അതിനെ ഒരു കാരക്റ്റർ അറെയിൽ സംഭരിക്കുന്നതിനുമുള്ള കൺസോൾ ഇൻപുട്ട് ഫങ്ഷനാണ് `gets()`. ഈ ഫങ്ഷനിലേക്ക് സ്ട്രിങ് വേരിയബിൾ (കാരക്റ്റർ അറെയുടെ പേര്) താഴെ കാണുന്നവിധത്തിൽ നൽകാവുന്നതാണ്.

```
gets(character_array_name);
```

ഈ ഫങ്ഷൻ ഉപയോഗിക്കുമ്പോൾ cstdio(stdio.h എന്നത് Turbo C++ൽ) എന്ന ലൈബ്രറി ഹെഡർ ഫയൽ പ്രോഗ്രാമിൽ ഉൾപ്പെടുത്തേണ്ടതാണ്. പ്രോഗ്രാം 9.1 ൽ include <cstdio> ഉൾപ്പെടുത്തുകയും കൂടാതെ cin>>my_name; എന്ന പ്രസ്താവനയ്ക്ക് പകരം gets(my_name); ഉപയോഗിച്ച് പ്രോഗ്രാം വീണ്ടും പ്രവർത്തിപ്പിച്ചാൽ താഴെ കാണുന്ന ഔട്ട്പുട്ട് ലഭിക്കുന്നതാണ്.

```
Enter your name : Maya Mohan

Hello Maya Mohan
```

ഇപ്പോൾ നാം ഇൻപുട്ട് ചെയ്ത മുഴുവൻ സ്ട്രിങ്ങും ഔട്ട്പുട്ട് ആയി കാണപ്പെടുന്നുണ്ട്. ഇനി നമുക്ക് gets() ഫങ്ഷനും cin ഉം തമ്മിലുള്ള വ്യത്യാസം എന്താണെന്നു നോക്കാം.

സ്ട്രിങ്ങിന്റെ ഇൻപുട്ട്/ഔട്ട്പുട്ട് പ്രവർത്തനങ്ങളിൽ സബ്സ്ക്രിപ്റ്റഡ് വേരിയബിൾ എന്ന ആശയം ഉപയോഗിക്കുന്നില്ലെങ്കിലും, അറെയിലെ ഏതൊരു അംഗത്തെയും അറെയുടെ പേരും സബ്സ്ക്രിപ്റ്റും ഉപയോഗിച്ചു വേർതിരിച്ചുപയോഗിക്കാവുന്നതാണ്. സ്ട്രിങ്ങിലെ ആദ്യത്തെ കാരക്റ്ററിനെ ഉപയോഗിക്കണമെങ്കിൽ my_name[0] എന്നും, അഞ്ചാമത്തെ കാരക്റ്റർ എടുത്തുപയോഗിക്കണമെങ്കിൽ my_name[4] എന്നിങ്ങനെ എന്നും പ്രയോഗിക്കാവുന്നതാണ്. നൾ കാരക്റ്ററും ('\0') നമുക്ക് സബ്സ്ക്രിപ്ട് ഉപയോഗിച്ചു തെരഞ്ഞെടുക്കാം. താഴെ കൊടുത്തിട്ടുള്ള പ്രോഗ്രാം ഈ ആശയം വ്യക്തമാക്കുന്നതാണ്.

പ്രോഗ്രാം 2.6 തന്നിരിക്കുന്ന സ്ട്രിങ്ങിലെ സ്വരാക്ഷരങ്ങളുടെ (Vowels) എണ്ണം കണ്ടുപിടിക്കുക

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
  {
   char str[20];
int vow=0;
   cout<<"Enter a string: ";
   gets(str);
   for(int i=0; str[i]!='\0'; i++)
      switch(str[i])
      {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u': vow++;
      }
   cout<<"No. of vowels in the string "<<str<<" is "<<vow;
   return 0;
  }
```

gets( ) ഫങ്ഷന് വേണ്ടിയുള്ള ഹെഡർ ഫയൽ

നൾ കാരക്ടർ എത്തുന്നതുവരെ തുടർന്നു കൊണ്ടിരിക്കുന്നു.

അറെയിലെ ഓരോ കാരക്ടറും കാരക്ടർ കോൺസ്റ്റന്റുമായി താരതമ്യം ചെയ്യുന്നു

"Hello guys" എന്ന സ്ട്രിങ് ഇൻപുട്ട് ചെയ്ത് പ്രോഗ്രാം 9.2 പ്രവർത്തിപ്പിക്കുകയാണെങ്കിൽ ചുവടെ കൊടുത്തിരിക്കുന്ന ഔട്ട്പുട്ട് കാണാവുന്നതാണ് .

```
Enter a string : Hello guys
No.of vowels in the string Hello guys is 3
```

ഈ പ്രോഗ്രാം പ്രവർത്തിച്ച് ഫലം ലഭ്യമാകുന്നത് എങ്ങനെയെന്ന് നമുക്ക് വിശകലനം ചെയ്യാം.

- തുടക്കത്തിൽ തന്നെ `gets()` ഫങ്ഷൻ ഉപയോഗിച്ച് `"Hello guys"` എന്ന സ്ട്രിങ് ഇൻപുട്ട് ചെയ്യുന്നു .
- 'i' എന്ന സബ്സ്ക്രിപ്റ്റ് ഉപയോഗിച്ചു സൂചിപ്പിക്കുന്ന അറെയിലെ ഓരോ കാരക്റ്ററും, നൾ കാരക്റ്റർ ('\0') അല്ലാത്തിടത്തോളം ഫോർ ലൂപ്പിന്റെ ചട്ടക്കൂട് തുടർച്ചയായി പ്രവർത്തിച്ചുകൊണ്ടിരിക്കുന്നു. അതായത് നൾ കാരക്റ്റർ എത്തുന്നതുവരെ ലൂപ്പിന്റെ ചട്ടക്കൂട് പ്രവർത്തിച്ചുകൊണ്ടിരിക്കും.
- ലൂപ്പ് ചട്ടക്കൂടിനകത്ത് ഒരേയൊരു സ്വിച്ച് പ്രസ്താവന (switch statement) മാത്രമേ ഉള്ളൂ. ആദ്യത്തെ നാലു കേസുകളിലും ഒരു പ്രസ്താവന പോലും നൽകിയിട്ടില്ല. അവസാനത്തെ കേസിന് vow എന്ന വേരിയബിളിന്റെ വില ഒന്ന് വർദ്ധിക്കുന്നു (vow++). എല്ലാ കേസുകൾക്കും ഇതാവശ്യമാണെന്നു ഒരു പക്ഷെ നിങ്ങൾ ചിന്തിക്കുന്നുണ്ടാവും. അത് തികച്ചും ശരിയാണ്. എന്നാൽ അങ്ങനെയാണെങ്കിൽ ഓരോ കേസിനും വെവ്വേറെ ബ്രേക്ക് പ്രസ്താവനകൾ ഉപയോഗിക്കേണ്ടതായി വരും. ഈ പ്രോഗ്രാമിൽ എല്ലാ കേസുകളുടെയും പ്രവർത്തനം ഒരേ പോലെയായതിനാലാണ് ഈ രീതിയിലുള്ള പ്രസ്താവന ഉപയോഗിച്ചിരിക്കുന്നത്.
- ഫോർ ലൂപ്പ് തുടർച്ചയായി പ്രവർത്തിക്കുമ്പോൾ ഓരോ കാരക്റ്ററും ഒന്നിന് പുറകെ ഒന്നായി ലഭ്യമാകുന്നു. അവയെ കേസിലെ ഓരോ കാരക്റ്റർ കോൺസ്റ്റന്റുമായി താര തമ്യം ചെയ്യുന്നു. ഏതെങ്കിലും ഒരു തവണ സമാനത കൈവരിച്ചാൽ vow എന്ന വേരിയബിളിന്റെ വില ഒന്ന് കൂടുന്നു (vow ++).
- നൽകിയിട്ടുള്ള ഇൻപുട്ട് സ്ട്രിങ്ങിന്റെ കാര്യത്തിൽ സമാനത കൈവരിക്കുന്നത് " i " യുടെ വില 1, 4, 7 എന്നിങ്ങനെ ആകുമ്പോഴാണ്. അതുകൊണ്ട് തന്നെ vow ന്റെ വില മൂന്നു തവണ ഓരോന്ന് വെച്ച് വർധിക്കുകയും നമുക്ക് ശരിയായ ഉത്തരം ലഭിക്കുകയും ചെയ്യുന്നു.

സ്ട്രിങ്ങുകൾ ഇൻപുട്ട് ചെയ്യുന്നതിനു `gets()` ഫങ്ഷൻ എങ്ങനെ ഉപയോഗിക്കുന്നുവെന്ന് നാം മനസ്സിലാക്കി. അതുപോലെ സ്ട്രിങ് ഔട്ട്പുട്ട് ചെയ്യുന്നതിന് C++ ൽ `puts()` എന്ന ഫങ്ഷൻ ലഭ്യമാണ്. സ്ട്രിങ് ഡാറ്റയെ സ്റ്റാൻഡേർഡ് ഔട്ട്പുട്ട് ഉപകരണ (മോണിറ്റർ) ത്തിൽ പ്രദർശിപ്പിക്കുവാൻ വേണ്ടിയുള്ള കൺസോൾ ഔട്ട്പുട്ട് ഫങ്ഷനാണ് `put()`. ഇതിന്റെ വാക്യഘടന (syntax) താഴെ കൊടുത്തിരിക്കുന്നു .

```
puts(string data);
```

ഈ ഫങ്ഷനിലേക്ക് സ്ട്രിങ് കോൺസ്റ്റന്റ് അഥവാ വേരിയബിൾ (കാരക്റ്റർ അറെയുടെ പേര്) ആണ് നൽകേണ്ടത്. താഴെ കാണുന്ന C++ കോഡ് നിരീക്ഷിക്കുക .

```
char str[10] ="friends";
puts("hello");
puts(str);
```

മേൽ സൂചിപ്പിച്ച കോഡിന്റെ ഔട്ട്പുട്ട് താഴെ കാണും വിധത്തിലാണ് .

```
hello
friends
```

കാരക്റ്റർ അറെ str[10] ലെ "friends" എന്ന സ്ട്രിങ് അടുത്ത ലൈനിലാണ് പ്രദർശിപ്പിച്ചിരിക്കുന്നത്. `puts()` ഫങ്ഷനുകൾക്ക് പകരം `cout<<"hello";, cout<< str;` എന്നീ പ്രസ്താവനകൾ ഉപയോഗിക്കുമ്പോഴുള്ള വ്യത്യാസം ശ്രദ്ധിക്കുക. cout ഉപയോഗിക്കുമ്പോൾ സ്ട്രിങ്ങുകൾക്കിടയിൽ ഒരു സ്പേസ് പോലും ഇല്ലാതെ ഔട്ട്പുട്ട് അതേ വരിയിൽ തന്നെ പ്രദർശിപ്പിക്കപ്പെടുന്നു.

നമുക്കു ചെയ്യാം

പ്രോഗ്രാം 2.6 ൽ "HELLO GUYS" എന്ന ഇൻപുട്ട് നൽകി ഔട്ട്പുട്ട് പ്രവചിക്കുക. പ്രോഗ്രാം പ്രവർത്തിപ്പിച്ചു ഈ ഇൻപുട്ടിന് ശരിയായ ഔട്ട്പുട്ട് ലഭിക്കുന്നുണ്ടോ എന്ന് പരിശോധിക്കുക. ഔട്പുട്ടിലുണ്ടായിരിക്കുന്ന വ്യത്യാസത്തിന് കാരണം കണ്ടെത്തുക. തന്നിരിക്കുന്ന ഏതൊരു സ്ട്രിങ്ങിനും അനുസരിച്ച് കൃത്യമായ ഔട്ട്പുട്ട് ലഭിക്കുന്നതിന് പ്രോഗ്രാമിൽ ആവശ്യമായ മാറ്റങ്ങൾ വരുത്തുക.

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 അംഗങ്ങളെയും സൂചികയുടെ സഹായത്താൽ ഉപയോഗിക്കുവാൻ കഴിയും. അറെയിലെ മെമ്മറി നീക്കിവയ്പ് ചിത്രങ്ങളുടെ സഹായത്താൽ വിശദീകരിച്ചിട്ടുണ്ട്. ലൂപ്പിങ്ങ് പ്രസ്താവനകൾ പ്രത്യേകിച്ചും അറെ അംഗങ്ങളെ മാറ്റുന്നതിനായുള്ളവ പ്രോഗ്രാമുകളുടെ സഹായത്തോടെ വിശദീകരിച്ചിട്ടുണ്ട്. പ്രോഗ്രാമുകളിൽ അറെ ഉപയോഗിച്ച് സ്ട്രിങ്ങുകൾ എങ്ങനെ കാര്യക്ഷമമായി കൈകാര്യം ചെയ്യാമെന്നും നാം പഠിച്ചു.

നമുക്ക് പരിശീലിക്കാം

1. Sales Amt എന്ന അറെയിലേക്ക് 12 മാസത്തെ വിറ്റ്വരവിന്റെ തുക ഇൻപുട്ട് നൽകുവാനുള്ള C++ പ്രോഗ്രാം എഴുതുക. എല്ലാ ഇൻപുട്ടും നൽകിയ ശേഷം ആകെ വിറ്റ് വരവിന്റെ തുകയും ശരാശരിയും കണക്കാക്കുക.
2. N അക്കങ്ങൾ ഉള്ള ഒരു അറെ നിർമിക്കുവാനുള്ള C++ പ്രോഗ്രാം എഴുതുക, ഇതിന്റെ ശരാശരി കണ്ടുപിടിച്ച് ശരാശരിയിൽ കൂടുതൽ ഉള്ള അക്കങ്ങൾ പ്രദർശിപ്പിക്കുക.
3. ഒരു പൂർണസംഖ്യ അറെയുടെ ആദ്യത്തേയും അവസാനത്തേയും അംഗങ്ങളെ ഒത്തു മാറ്റുവാൻ (swap) ഉള്ള C++ പ്രോഗ്രാം എഴുതുക.
4. പത്ത് പൂർണ സംഖ്യകളുടെ ഒരു അറെയിൽ സ്വീകരിക്കുവാനുള്ള C++ പ്രോഗ്രാം എഴുതുകയും ഇവയിൽ നിന്ന് ഏറ്റവും കൂടിയ സംഖ്യയും, കൂടിയ സംഖ്യയും കണ്ടെത്തുക.
5. ഒരു വാചകം അഥവാ സ്ക്രീംഗ് സ്വീകരിക്കുവാനുള്ള ഒരു C++ പ്രോഗ്രാം എഴുതുകയും വലിയ അക്ഷരങ്ങൾ, ചെറിയ അക്ഷരങ്ങൾ, സംഖ്യകൾ, പ്രത്യേക ചിഹ്നങ്ങൾ, ശൂന്യ സ്ഥലം തുടങ്ങിയവയുടെ എണ്ണം കണ്ടെത്തുക.
6. ഒരു വാചകത്തിന്റെ വാക്കുകളുടെ എണ്ണം കണ്ടെത്തുവാനുള്ള C++ പ്രോഗ്രാം എഴുതുക.
7. ഒരു സ്ട്രിംഗിന്റെ അളവ് കണ്ടെത്തുവാനുള്ള C++ പ്രോഗ്രാം എഴുതുക.

നമുക്ക് വിലയിരുത്താം

1. പത്ത് അംഗങ്ങളുള്ള ഒരു അറെയിലെ അംഗങ്ങളെ എണ്ണപ്പെടുന്നത് ..... മുതൽ ..... വരെയാണ്.
2. ഒരു അറെയിലെ അംഗത്തെ ഉപയോഗിക്കുന്നത് ..... ആധാരമാക്കിയാണ്.
3. AR എന്നത് ഒരു അറെയാണ് എങ്കിൽ, AR[7] എന്നത് അറെയിലെ ഏത് അംഗത്തേ പ്രതിനിധീകരിക്കുന്നു.
4. int a[3] = {2,3,4}; എന്ന അറെയുടെ പ്രഖ്യാപനം പരിഗണിക്കുക. a[1] എന്നതിന്റെ വില എന്താണ്?
5. int a[ ] = {1,2,4} എന്ന അറെയുടെ പ്രഖ്യാപനം പരിഗണിക്കുക. എങ്കിൽ a[1] എന്നതിന്റെ വില എന്താണ്?
6. അറെയിലെ എല്ലാ അംഗങ്ങളെയും അച്ചടിക്കുക എന്നത് ..... പ്രവർത്തനത്തിന് ഉദാഹരണമാണ്.

7. ചുവടെ തന്നിരിക്കുന്ന കോഡ് ശകലത്തിന്റെ ഔട്ട്പുട്ട് എഴുതുക.

```
puts ("hello");
puts ("friends");
```

8. "GCC" എന്ന വാക്ക് സൂക്ഷിക്കുന്നതിനായുള്ള പ്രാരംഭ വില നൽകൽ പ്രസ്താവന എഴുതുക.

9. അറെ നിർവചക്കുക.

10. int studlist[1000] എന്ന പ്രഖ്യാപനം കൊണ്ട് ഉദ്ദേശിക്കുന്നത് എന്താണ്?

11. എങ്ങനെയാണ് ഏക ത്രിമാനമായ/ഡയമെൻഷണൽ ആയ അറെയ്ക്കായി മെമ്മറി നീക്കി വയ്ക്കുന്നത്.

12. പത്ത് അംഗങ്ങളെ സ്വീകരിക്കുന്നതിനായിട്ടുള്ള C++ പ്രസ്താനവകൾ എഴുതുകയും ഒറ്റ സംഖ്യകളുടെയും ഇരട്ട സംഖ്യകളുടെ എണ്ണം പ്രദർശിപ്പിക്കുക.

13. ചുവടെ ചേർത്തിരിക്കുന്ന പ്രസ്താവനകൾ വായിക്കുക.

```
char namep[20]
cin>>name;
cout<<name;
```

നിങ്ങൾ "sachin tendulkar" എന്ന വാചകമാണ് ഇൻപുട്ടായി നൽകുന്നത് എങ്കിൽ എന്താകും ഇതിന്റെ ഔട്ട്പുട്ട്. നിങ്ങളുടെ ഉത്തരത്തെ സാധൂകരിക്കുക.

14. ഒരേ വലിപ്പമുള്ള രണ്ട് ഏക ഡയമെൻഷണൽ അറെ സ്വീകരിക്കുവാനുള്ള പ്രസ്താവനകൾ എഴുതുകയും തത്തുല്യമായ അംഗങ്ങൾ തമ്മിലുള്ള വ്യത്യാസം കണ്ടെത്തുകയും ചെയ്യുക.

15. ഒരു വാചകം അല്ലെങ്കിൽ ഒരു വാക്ക് അനുലോമവിലോമ പദം (palindrome) ആണോ എന്ന് പരിശോധിക്കുവാനുള്ള പ്രോഗ്രാം എഴുതുക.