9

സ്ട്രിങ് കൈകാര്യം ചെയ്യലും ഇൻപുട്ട് /ഔട്ട്പുട്ട് ഫങ്ഷനുകളും

പ്രധാന ആശയങ്ങൾ

- അറെ ഉപയോഗിച്ചുള്ള സ്ട്രിങ് കൈകാര്യം ചെയ്യൽ
- സ്ട്രിങ്ങിനു വേണ്ടിയുള്ള മെമ്മറി നീക്കിവെയ്ക്കൽ
- സ്ട്രിങ്ങിനു മേലുള്ള ഇൻപുട്ട് /ഔട്ട്പുട്ട് പ്രവർത്തനങ്ങൾ
- കാരക്റ്റർ ഇൻപുട്ട് /ഔട്ട്പുട്ട് പ്രവർത്തനങ്ങൾക്ക് വേണ്ടിയുള്ള കൺസോൾ ഫങ്ഷനുകൾ
  - getchar()
  - putchar()
- ഇൻപുട്ട് /ഔട്ട്പുട്ട് പ്രവർത്തനങ്ങൾക്കുള്ള സ്ട്രീം ഫങ്ഷനുകൾ
  - ഇൻപുട്ട് ഫങ്ഷനുകൾ get(), getline()
  - ഔട്ട്പുട്ട് ഫങ്ഷനുകൾ put(), write()

ഒരേ തരത്തിലുള്ള അനേകം ഡാറ്റയെ കൈകാര്യം ചെയ്യുന്നതിനുള്ള ഫലപ്രദമായ ഉപാധിയാണ് അറേകൾ (Arrays) എന്ന് നാം പഠിച്ചു കഴിഞ്ഞു. ഇതിനു മുമ്പ് ചർച്ച ചെയ്ത മിക്കവാറും പ്രോഗ്രാമുകളിലും അറേകൾ ഉപയോഗിച്ചിരിക്കുന്നത് ന്യൂമെറിക് ഡാറ്റ ഇനങ്ങളെ പ്രോസസ്സ് ചെയ്യുന്നതിനാണ്. എന്നാൽ സ്ട്രിങ് ടൈപ്പ് ഡാറ്റയും ഉണ്ടെന്നത് നമുക്കറിയാവുന്ന ഒരു വസ്തുതയുമാണ്. അത്തരം ഡാറ്റയെ മെമ്മറിയിൽ ശേഖരിക്കുന്നതും പ്രോസസ്സ് ചെയ്യുന്നതും നാം ഇവിടെ ചർച്ചചെയ്യുന്നു. കൂടാതെ സ്ട്രിങ്ങുകളെയും കാരക്റ്ററുകളെയും കൈകാര്യം ചെയ്യുന്നതിനുള്ള ചില ഇൻപുട്ട്/ഔട്ട്പുട്ട് അന്തർനിർമിത ഫങ്ഷനുകളും (Built in functions) ഇവിടെ പ്രതിപാദിക്കപ്പെടുന്നുണ്ട്.

9.1. 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? അതു

കൊണ്ട് നമുക്ക് char ഡാറ്റ ഇനത്തെക്കുക്കുറിച്ച് ആലോചിക്കാം. എന്നാൽ അവിടെയും ഒരു പ്രശ്നമുണ്ട്. char ഡാറ്റ ഇനത്തിന് ഒരു കാരക്റ്റർ മാത്രമേ ശേഖരിക്കുവാൻ കഴിയുകയുള്ളു. 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];
```

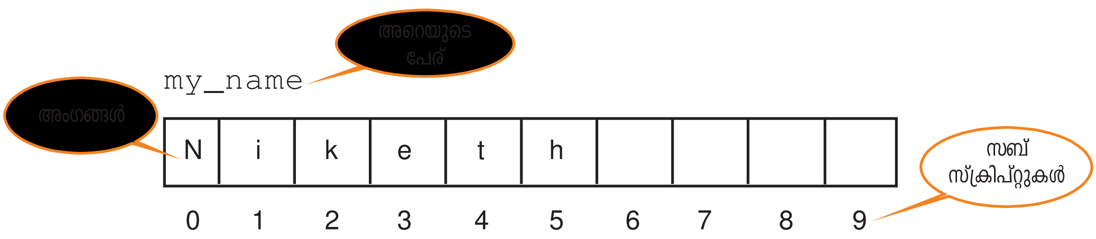
my_name എന്ന് പേരുള്ള അറേയിൽ ഒരു ബൈറ്റ് വീതം വലിപ്പമുള്ള തുടർച്ചയായ പത്ത് മെമ്മറി സ്ഥാനങ്ങൾ നീക്കിവച്ചിട്ടുണ്ട്. ഈ അറേയിലേക്ക് താഴെ കാണുന്നത് പോലെ പ്രാരംഭ വിലകൾ നൽകാവുന്നതാണ്.

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

ചിത്രം 9.1ൽ 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 9.1 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 9.1 ൽ പറഞ്ഞിരിക്കുന്നത് പോലെ ഇത് ചെയ്യാവുന്നതാണ് .

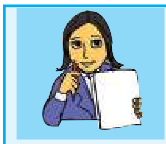
പ്രോഗ്രാം 9.1: ഒരു സ്ട്രിങ് ഇൻപുട്ട് ചെയ്ത് പ്രദർശിപ്പിക്കുക.

```
     #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).



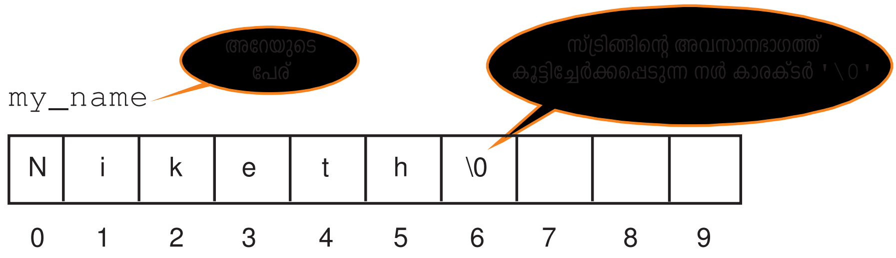
നമുക്ക് ചെയ്യാം

പ്രോഗ്രാം 9.1 പ്രവർത്തിപ്പിച്ച് നിങ്ങളുടെ പേരിന്റെ കൂടെ ഇനിഷ്യലും ഇൻപുട്ട് ചെയ്ത് ഔട്ട്പുട്ട് ശരിയോ തെറ്റോ എന്ന് പരിശോധിക്കുക. പേരിൽ 10 കാരക്റ്ററുകളിലും കൂടുതൽ ഉണ്ടെങ്കിൽ അറെയുടെ വലിപ്പം ആവശ്യത്തിനനുസരിച്ച് വർദ്ധിപ്പിക്കുക .

9.2 സ്ട്രിങ്ങിനു വേണ്ടിയുള്ള മെമ്മറി നീക്കിവെയ്പ് (Memory allocation for strings)

ഒരു അറെയിലുള്ള കാരക്റ്ററുകൾക്ക് എങ്ങനെയാണ് മെമ്മറി അനുവദിക്കുന്നതെന്നു നാം കണ്ടു കഴിഞ്ഞു. ചിത്രം 9.1 ൽ കാണിച്ചിരിക്കുന്നത് പോലെ മെമ്മറി ആവശ്യകത കണക്കാക്കുന്നത് ഇൻപുട്ട് ചെയ്ത കാരക്റ്ററുകളുടെ എണ്ണമനുസരിച്ചാണ്. എന്നാൽ ഒരു

കാരക്റ്റർ അറെയിൽ സ്ട്രിങ് ഇൻപുട്ട് ചെയ്യുമ്പോൾ ചിത്രം മറ്റൊന്നാകുന്നു. നമ്മൾ പ്രോഗ്രാം 9.1 പ്രവർത്തിപ്പിച്ച് "Niketh" എന്ന സ്ട്രിങ് ഇൻപുട്ട് ചെയ്താൽ മെമ്മറി നീക്കിവെയ്പ് താഴെ കാണുന്ന വിധമായിരിക്കും.



ചിത്രം 9.2 : കാരക്റ്റർ അറേയുടെ 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 .

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

പ്രോഗ്രാം 9.1ൽ 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 (>>) മാത്രമേയുള്ളൂ. 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 സബ്സ്ക്രിപ്റ്റ് ഉപയോഗിച്ചു തെരഞ്ഞെടുക്കാം. താഴെ കൊടുത്തിട്ടുള്ള പ്രോഗ്രാം ഈ ആശയം വ്യക്തമാക്കുന്നതാണ്.

പ്രോഗ്രാം 9.2 തന്നിരിക്കുന്ന സ്ട്രിങ്ങിലെ സ്വരാക്ഷരങ്ങളുടെ (Vowels) എണ്ണം കണ്ടുപിടിക്കുക

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
 {
  char str[20];
```

gets( ) ഫങ്ഷന് വേണ്ടിയുള്ള ഹെഡർ ഫയൽ

```
 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;
}
```

നൾ കാരക്ടർ എത്തുന്നതുവരെ തുടർന്നു കൊണ്ടിരിക്കുന്നു.

അറേയിലെ 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 താരതമ്യം 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.

നമുക്ക് ചെയ്യാം

പ്രോഗ്രാം 9.2 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.

9.4 കാരക്റ്റർ ഇൻപുട്ട്/ഔട്ട്പുട്ട് നുവേണ്ടിയുള്ള കൺസോൾ ഫങ്ഷനുകൾ (More console functions)

സ്ട്രിങ്ങിന് മേലുള്ള ഇൻപുട്ട്/ഔട്ട്പുട്ട് പ്രവർത്തനങ്ങൾ നാം ചർച്ച ചെയ്ത് കഴിഞ്ഞു. കാരക്റ്ററുകൾക്ക് മേൽ പ്രയോഗിക്കുവാനുള്ള ചില ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഫങ്ഷനുകളും C++ ൽ ഉൾപ്പെടുത്തിയിട്ടുണ്ട്. ഇത്തരം ഫങ്ഷനുകൾ ഉപയോഗിക്കുന്നതിന് **cstdio**

(stdio.h എന്നത് Turbo C++ ൽ) എന്ന ഹെഡർ ഫയൽ പ്രോഗ്രാമിൽ ഉൾപ്പെടുത്തേണ്ടത് അത്യാവശ്യമാണ് .

getchar()

ഈ ഫങ്ഷൻ കീബോർഡിലൂടെ ഇൻപുട്ട് ചെയ്ത കാരക്റ്ററിനെ തിരികെ തരികയാണ് ചെയ്യുന്നത്. താഴെ കൊടുത്തിട്ടുള്ള ഉദാഹരണത്തിൽ കാണുന്ന പോലെ ഒരു കാരക്റ്ററിനെ വേരിയബിലിളിലേക്ക് സംഭരിക്കാവുന്നതാണ്.

```
char ch=getchar();
```

സ്ട്രിങ് ഔട്ട്പുട്ടിൽ puts() ഫങ്ഷന്റെ മേന്മകൾ നാം കണ്ടു കഴിഞ്ഞു. ഇനി നമുക്ക് കാരക്റ്റർ ഡാറ്റ ഔട്ട്പുട്ടായി ലഭിക്കുവാനുള്ള ഫങ്ഷനെക്കുറിച്ച് പഠിക്കാം.

putchar()

തന്നിരിക്കുന്ന കാരക്റ്റർ ആർഗ്യുമെന്റിനെ സ്റ്റാൻഡേർഡ് ഔട്ട്പുട്ട് ഉപകരണ (മോണിറ്റർ) ത്തിൽ പ്രദർശിപ്പിക്കുകയാണ് ഈ ഫങ്ഷൻ ചെയ്യുന്നത്. ഇവിടെ ആർഗ്യുമെന്റ് ഒരു കാരക്റ്റർ കോൺസ്റ്റന്റോ അല്ലെങ്കിൽ ഒരു വേരിയബിളോ ആവാം. ആർഗ്യുമെന്റായി ഒരു പൂർണ സംഖ്യയാണ് (integer) നൽകുന്നതെങ്കിൽ അതിനെ ഒരു ASCII വിലയായി പരിഗണിക്കുകയും അതിനുനസൃതമായ കാരക്റ്റർ പ്രദർശിപ്പിക്കുകയും ചെയ്യുന്നു. താഴെ കൊടുത്തിട്ടുള്ള കോഡ് putchar() ഫങ്ഷന്റെ ഉപയോഗം വ്യക്തമാക്കുന്നു.

```
char ch='B';   // വേരിയബിൾ ch നകത്ത് 'B' ശേഖരിക്കപ്പെടുന്നു
putchar(ch);   //'B' സ്ക്രീനിൽ പ്രദർശിപ്പിക്കപ്പെടുന്നു
ptchar('c');   // 'c' സ്ക്രീനിൽ പ്രദർശിപ്പിക്കപ്പെടുന്നു
putchar(97);   // 97 എന്ന ASCII വിലയ്ക്കനുസൃതമായ 'a' സ്ക്രീനിൽ
               പ്രദർശിപ്പിക്കപ്പെടുന്നു.
```

പോഗ്രാം 9 .3 ഈ ഫങ്ഷനുകളുടെ പ്രവർത്തനം വ്യക്തമാക്കുന്നതാണ്. ഒരു സ്ട്രിങ് ഇൻപുട്ട് ചെയ്ത് ഒരു കാരക്റ്റർ കണ്ടെത്തുവാൻ ഈ പ്രോഗ്രാമിലൂടെ സാധിക്കുന്നു. ഒരു കാരക്റ്റർ എത്ര തവണ ആവർത്തിക്കുന്നുവെന്നു പ്രദർശിപ്പിക്കുകയാണ് ഈ പ്രോഗ്രാം ചെയ്യുന്നത് .

പ്രോഗ്രാം 9.3 തന്നിരിക്കുന്ന കാരക്റ്റർ ഒരു സ്ട്രിങ്ങിനകത്ത് എത്ര തവണ ഉണ്ടെന്നു കൺസോൾ ഫങ്ഷൻ ഉപയോഗിച്ച് കണ്ടെത്തുക

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
   char str[20], ch;
   int i, num=0;
   puts("Enter a string:"); //To print '\n' after the string
```

```
    gets(str);//To accept a string with white spaces
    cout<<"Enter the character to be searched: ";
    ch=getchar(); //To input the character to be searched
    /* A loop to search for the character and count its
       occurrences in the string. Search will be
       terminated when a null character is found   */
    for(i=0; str[i]!='\0'; i++)
        if (str[i]==ch)
            num++;
    cout<<"The string \'"<<str<<"\' uses the character \'";
    putchar(ch);
    cout<<"\' ")<<num<<" times";
    return 0;
}
```

ഇതുവരെ ചർച്ച ചെയ്ത എല്ലാ കൺസോൾ ഫങ്ഷനുകളും പ്രോഗ്രാം 9.3 യിൽ ഉപയോഗിച്ചിട്ടുണ്ട്. ഈ പ്രോഗ്രാമിന്റെ ഒരു മാതൃക ഔട്ട്പുട്ട് താഴെ കൊടുത്തിരിക്കുന്നു.

Enter the string :

I have a dream

Enter the character to be searched : a

The string "I have a dream" uses the character 'a' 3 times

സ്വയം പരിശോധിക്കാം

1. ഒരു സ്ട്രിങ്ങിന്റെ അവസാനം സൂചിപ്പിക്കാൻ മെമ്മറിയിൽ ഉപയോഗിക്കുന്ന കാരക്റ്റർ ഏത്?
2. 'Save earth' എന്ന സ്ട്രിങ്ങ് ശേഖരിക്കുന്നതിനുള്ള വേരിയബിൾ പ്രഖ്യാപന പ്രസ്താവന എഴുതുക?
3. കൺസോൾ ഇൻപുട്ട് / ഔട്ട്പുട്ട് ഉപയോഗിക്കുന്നതിനാവശ്യമായ ഹെഡർ ഫയലിന്റെ പേരെഴുതുക?
4. 'Be Positive' എന്ന സ്ട്രിങ്ങ് ശേഖരിക്കുന്നതിനു എത്ര ബൈറ്റുകൾ ആവശ്യമാണ്?
5. `puts ("hello"); cout<<"hello";` എന്നിവ എങ്ങനെ വ്യത്യാസപ്പെട്ടിരിക്കുന്നു?

9.5 ഇൻപുട്ട് / ഔട്ട്പുട്ട് പ്രക്രിയകൾക്ക് വേണ്ടിയുള്ള സ്ട്രീം ഫങ്ഷനുകൾ (Stream functions for I/O operations)

കാരക്റ്ററുകളിലും സ്ട്രിങ്ങുകളിലും ഇൻപുട്ട്/ഔട്ട്പുട്ട് പ്രക്രിയകൾ ചെയ്യുവാനുള്ള മറ്റൊരു സൗകര്യം C++ ൽ ലഭ്യമാക്കിയിട്ടുണ്ട്. iostream എന്ന ഹെഡർ ഫയലിൽ ഉൾപ്പെടുത്തിയിട്ടുള്ള ഫങ്ഷനുകളാണിവ. മെമ്മറിക്കും ഒബ്ജക്റ്റുകൾക്കുമിടയിൽ പ്രവഹിക്കുവാൻ ബൈറ്റുകളെ (ഡാറ്റ) (stream of bytes) യെ കൈകാര്യം ചെയ്യുന്നതിനാൽ ഇവയെ പൊതുവെ സ്ട്രീം ഫങ്ഷനുകൾ എന്നാണ് വിളിക്കുന്നത്. C++ൽ കീബോർഡ്, മോണിറ്റർ

എന്നിവയെയാണ് സാധാരണയായി ഒബ്ജക്റ്റുകളായി സൂചിപ്പിച്ചിരിക്കുന്നത്. ഇവയിൽ ഏതാനും ചില ഫങ്ഷനുകൾ നമുക്ക് പരിശോധിക്കാം .

A. ഇൻപുട്ട് ഫങ്ഷനുകൾ

കാരക്റ്റർ /സ്ട്രിങ് ഡാറ്റയെ ഇൻപുട്ട് ചെയ്യുന്നതിന് ഉപയോഗിക്കുന്ന ഫങ്ഷനുകളാണിവ. ഒബ്ജക്റ്റുകൾക്കും മെമ്മറിക്കുമിടയിൽ ബൈറ്റുകളെ പ്രവഹിക്കുവാൻ സഹായിക്കുന്ന ഫങ്ഷനുകളാണ് `get()`, `getline()` എന്നിവ. കീ ബോർഡ് ഉപയോഗിച്ച് ഡാറ്റ ഇൻപുട്ട് ചെയ്യുമ്പോൾ കീബോർഡിനെ സൂചിപ്പിക്കാൻ cin എന്ന ഓബ്ജക്റ്റ് ഉപയോഗിക്കുകയും മേൽപ്പറഞ്ഞ ഫങ്ഷനുകൾ `cin.get()`, `cin.getline()` എന്നീ രീതികളിൽ വിളിക്കുകയോ പ്രയോഗക്ഷമമാക്കുകയോ ചെയ്യുന്നു. ഇവിടെ ഡോട്ട് ഓപ്പറേറ്റർ എന്ന് വിളിക്കുന്ന പീരിയേഡ് (period) ചിഹ്നം (.) ആണ് `cin` എന്ന ഒബ്ജെക്റ്റിനും ഫങ്ഷനുമിടയിൽ ഉപയോഗിച്ചിരിക്കുന്നത്.

i. get()

കീബോർഡിലൂടെ ഒരു കാരക്റ്ററിനെയോ ഒന്നിലധികം കാരക്റ്ററുകളെയോ സ്വീകരിക്കുവാൻ ഈ ഫങ്ഷൻ ഉപയോഗിക്കുന്നു. ഒരു സ്ട്രിങ്ങിനെ സ്വീകരിക്കുന്നതിന് ഫങ്ഷന്റെ ആർഗ്യുമെന്റായി അറേയുടെ പേരും വലിപ്പവും നൽകേണ്ടതാണ്. താഴെ കൊടുത്തിരിക്കുന്ന കോഡ് ഈ ഫങ്ഷന്റെ ഉപയോഗം വ്യക്തമാക്കുന്നതാണ്.

```
char ch,str[10];
ch = cin.get(ch); //  ഒരു കാരക്റ്റർ സ്വീകരിച്ച് 'ch' ൽ ശേഖരിക്കുന്നു.
cin.get(ch);        // മേൽ സൂചിപ്പിച്ച പ്രസ്താവനയ്ക് സമാനം.
cin.get(str,10); // പരമാവധി 10 കാരക്റ്ററുകളുള്ള സ്ട്രിങ്ങിനെ  സ്വീകരിക്കുന്നു.
```

ii. getline()

കീബോർഡിലൂടെ ഒരു സ്ട്രിങ്ങിനെ സ്വീകരിക്കുവാനുള്ള ഫങ്ഷനാണിത്. എന്റർ കീ, കാരക്റ്ററുകളുടെ എണ്ണം അല്ലെങ്കിൽ ഏതെങ്കിലും പ്രത്യേക കാരക്റ്റർ, ഇവയിൽ ഏതെങ്കിലും ഉപയോഗിച്ചാണ് സ്ട്രിങ്ങിന്റെ അവസാനം സൂചിപ്പിക്കുന്നത്. ഈ ഫങ്ഷൻ ഉപയോഗിക്കുന്നതിനുള്ള രണ്ടുതരം വാക്യഘടന താഴെ കൊടുക്കുന്നു.

```
char ch,str[10];
int len;
cin.getline(str,len);      //  2 ആർഗ്യുമെന്റുകൾ  സഹിതം.
cin.getline(str,len,ch);  //  3 ആർഗ്യുമെന്റുകൾ  സഹിതം .
```

ആദ്യത്തേതിൽ `getline()` ഫങ്ഷന് രണ്ട് ആർഗ്യുമെന്റുകളായ കാരക്റ്റർ അറേയും (ഇവിടെ `str`) കൂടാതെ ആകെ എത്ര കാരക്റ്ററുകൾ ശേഖരിക്കാമെന്നു സൂചിപ്പിക്കുന്ന ഇന്റിജർ വിലയും (len) ഉണ്ട്. രണ്ടാമത്തേതിൽ സ്ട്രിങ്ങിന്റെ അവസാനം (Delimiter) സൂചിപ്പിക്കുന്ന കാരക്റ്ററും (`ch` വേരിയബിളിന്റെ വില) കൂടി ആകെ കാരക്റ്ററുകളുടെ എണ്ണത്തിനൊപ്പം നൽകിയിരിക്കുന്നു. സ്ട്രിങ് ഇൻപുട്ട് ചെയ്യുമ്പോൾ ഒന്നുകിൽ കാരക്റ്ററുകൾ മാത്രം (len-1) അല്ലെങ്കിൽ സ്ട്രിങ്ങിന്റെ അവസാനം സൂചിപ്പിക്കുന്ന കാരക്റ്റർ വരെ, ഇവയിലേതാണോ ആദ്യം സംഭവിക്കുന്നത് എന്നതിനെ ആശ്രയിച്ചായിരിക്കും സ്ട്രിങ് സംഭരിക്കപ്പെടുന്നത്.

B. ഔട്ട്പുട്ട് ഫങ്ഷനുകൾ

മെമ്മറിയ്ക്കും ഒബ്ജെക്റ്റിനുമിടയിൽ ഡാറ്റ ബൈറ്റുകൾ തുടർച്ചയായി പ്രവഹിക്കുവാൻ സഹായിക്കുന്ന ഔട്ട്പുട്ട് ഫങ്ഷനുകളാണ് put(), write() എന്നിവ. ഔട്ട്പുട്ട് ലഭിക്കുവാൻ വേണ്ടി മോണിറ്റർ ഉപയോഗിക്കുന്നതിനാൽ `cout` എന്ന ഒബ്ജക്റ്റ് ആണ് ഈ ഫങ്ഷനുകളുടെ കൂടെ ഉപയോഗിക്കുന്നത് .

i. put()

ഒരു കാരക്റ്റർ കോൺസ്റ്റന്റോ അല്ലെങ്കിൽ വേരിയബിളോ ആർഗ്യുമെന്റായി സ്വീകരിച്ചു പ്രദർശിപ്പിക്കുവാൻ ഉപയോഗിക്കുന്ന ഫങ്ഷനാണിത്.

```
char ch='c';
cout.put(ch);  // 'c' പ്രദർശിപ്പിക്കപ്പെടുന്നു.
cout.put('B'); // 'B' പ്രദർശിപ്പിക്കപ്പെടുന്നു.
cout.put(65);  // ASCII വില 65 നു പകരമായി 'A' പ്രദർശിപ്പിക്കപ്പെടുന്നു.
```

ii. write()

ആർഗ്യുമെന്റായി നൽകിയിട്ടുള്ള സ്ട്രിങ്ങിനെ പ്രദർശിപ്പിക്കുവാനാണ് ഇത് ഉപയോഗിക്കുന്നത്. വ്യക്തതയ്ക്ക് വേണ്ടി താഴെ കൊടുത്തിരിക്കുന്ന ഉദാഹരണം കാണുക.

```
char str[10] ="hello";
cout.write(str,10);
```

മേൽപ്പറഞ്ഞ കോഡ് പ്രവർത്തിപ്പിക്കുമ്പോൾ "hello" എന്ന സ്ട്രിങ്ങിന് ശേഷം 5 വൈറ്റ് സ്പേസോടു കൂടിയാണ് പ്രദർശിപ്പിക്കപ്പെടുന്നത്. കാരണം രണ്ടാമത്തെ ആർഗ്യുമെന്റിന്റെ വില 10 ഉം കൂടാതെ സ്ട്രിങ്ങിലെ ആകെ കാരക്റ്ററുകളുടെ എണ്ണം 5 ഉം ആയതിനാലാണ്.

പ്രോഗ്രാം 9.4 .സ്ട്രീം ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഫങ്ഷനുകളുടെ പ്രവർത്തനം വിശദമാക്കുന്നതിന്

```
#include <iostream>
#include <cstring> //To use strlen() function
using namespace std;
int main()
{
   char ch, str[20];
   cout<<"Enter a character: ";
   cin.get(ch); //To input a character to the variable ch
   cout<<"Enter a string: ";
   cin.getline(str,10,'.');     //To input the string
   cout<<"Entered character is:\t";
   cout.put(ch);             //To display the character
   cout.write("\nEntered string is:",20);
   cout.write(str,strlen(str));
   return 0;
}
```

പ്രോഗ്രാം 9.4 പ്രവർത്തിപ്പിക്കുമ്പോൾ താഴെ കാണുന്ന തരത്തിലുള്ള ഔട്ട്പുട്ട് ലഭ്യമാവുന്നതാണ്.

```
Enter a character: p
Enter a string: hello world
Entered character is:     p
Entered string is:
hello wo
```

ഈ പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുമ്പോൾ എന്താണ് സംഭവിക്കുന്നത് എന്ന് നോക്കാം. തുടക്കത്തിൽ തന്നെ `get()` ഫങ്ഷൻ `'p'` എന്ന കാരക്റ്ററിനെ സ്വീകരിക്കുന്നു. തുടർന്ന് `getline()` ഫങ്ഷൻ ഉപയോഗിച്ച് `"hello world"` എന്ന സ്ട്രിങ് ഇൻപുട്ട് ചെയ്യുന്നു. അതിന് ശേഷം `put()` ഫങ്ഷനുപയോഗിച്ച് `'p'` എന്ന കാരക്റ്റർ പ്രദർശിപ്പിക്കുന്നു. `write()` ഫങ്ഷൻ `"hello wo"` എന്ന് മാത്രമാണ് പ്രദർശിപ്പിക്കുന്നത്. `str` എന്ന അറേയിൽ ശേഖരിക്കാവുന്ന പാമാവധി കാരക്റ്ററുകളുടെ എണ്ണം 10 ആണെന്ന് ഫങ്ഷനിൽ സൂചിപ്പിച്ചിട്ടുമുണ്ട്. ഒരു ബൈറ്റ് നൾ കാരക്റ്ററിന് (`'\0'`– സ്ട്രിങ്ങിന്റെ അവസാന കാരക്റ്റർ) വേണ്ടി മാറ്റിവെക്കപ്പെട്ടതിനാൽ സാധാരണ 9 കാരക്റ്ററുകൾ മാത്രമേ ശേഖരിക്കുവാൻ കഴിയുകയുള്ളു. എന്നാൽ ഇവിടെ ഔട്ട്പുട്ട് ആയി വൈറ്റ് സ്പേസ് ഉൾപ്പെടെ 8 കാരക്റ്ററുകൾ മാത്രമാണ് കാണപ്പെടുന്നത്. ഇതിനു കാരണം `"p"` എന്ന കാരക്റ്ററിന് ശേഷം എന്റർ കീ ഉപയോഗിക്കുമ്പോൾ `'\n'`, `str` എന്ന അറേയുടെ ആദ്യത്തെ അംഗമായി ശേഖരിക്കപ്പെടുന്നതിനാലാണ്. അതുകൊണ്ട് `"hello wo"` എന്ന സ്ട്രിങ് പുതിയ വരിയിലാണ് പ്രദർശിപ്പിക്കപ്പെടുന്നത്.

ഈ പ്രോഗ്രാമിൽ `"hello.world"` എന്ന് ഇൻപുട്ട് ചെയ്ത് പ്രവർത്തിപ്പിച്ചാൽ താഴെ കാണുന്ന വിധത്തിലുള്ള ഔട്ട്പുട്ട് ലഭിക്കുന്നതാണ്.

```
Enter a character : a
Enter a string : hello.world
Entered character string is : a
Entered string is :
hello
```

(.) ഡോട്ട് കാരക്റ്റർ ശ്രദ്ധിക്കുക.

ഈ മാറ്റം ഔട്പുട്ടിൽ ഉണ്ടായതിന് കാരണം `getline()` എന്ന ഫങ്ഷൻ ഡോട്ട് ചിഹ്നത്തിന് `(dot operator)` മുമ്പുള്ള കാരക്റ്ററുകളെ മാത്രം സ്വീകരിച്ചതിനാലാണ്.

കീബോർഡിലെ കീകൾ ഉപയോഗിച്ച് ഇൻപുട്ട് ചെയ്യുമ്പോൾ ഒരുപാടു കാര്യങ്ങൾ സംഭവിക്കുന്നതിനാൽ ഇത്തരം ഫങ്ഷനുകൾ ഉപയോഗിക്കുമ്പോൾ ജാഗ്രത പുലർത്തേണ്ടതാണ്. അതുകൊണ്ടു തന്നെ നിങ്ങൾ ആഗ്രഹിക്കുന്ന ഔട്ട്പുട്ട് തന്നെ ലഭിക്കണമെന്നില്ല. മറ്റൊരു കാര്യം ശ്രദ്ധിക്കേണ്ടത് `write()` ഫങ്ഷനകത്ത് `strlen()` എന്ന ഫങ്ഷൻ ഉപയോഗിച്ചിരി

ക്കുന്നുവെന്നതാണ്. ഈ ഫങ്ഷൻ ഉപയോഗിക്കുന്നതിനു പകരം നേരിട്ട് 10 അല്ലെങ്കിൽ 20 എന്നീ സംഖ്യകൾ നൽകാവുന്നതാണ്. കൊടുത്തിരിക്കുന്ന സംഖ്യയെക്കാളും കുറവാണ് കാരക്റ്ററുകളുടെ എണ്ണമെങ്കിൽ ഇൻപുട്ട് ചെയ്യപ്പെട്ട സ്ട്രിങ്ങിന്റെ തുടർച്ചയായി ചില ASCII കാരക്റ്ററുകളും ചേർന്നാണ് ഔട്ട്പുട്ട് ലഭിക്കുക. നിങ്ങൾ `strlen()` ഉപയോഗിക്കുമ്പോൾ സ്ട്രിങ്ങിനകത്തെ കാരക്റ്ററുകളുടെ കൃത്യമായ എണ്ണം സൂചിപ്പിക്കപ്പെടുന്നുണ്ട്. ഈ ഫങ്ഷനുകളെ കുറിച്ചുള്ള കൂടുതൽ വിവരങ്ങൾ പത്താമത്തെ അധ്യായത്തിൽ ചർച്ച ചെയ്യാം. string.h എന്ന ഹെഡർ ഫയൽ ഉൾപ്പെടുത്തിയാൽ മാത്രമേ ഈ ഫങ്ഷൻ ഉപയോഗിക്കുവാൻ കഴിയുകയുള്ളൂ.

നമുക്ക് ചെയ്യാം

താഴെ കൊടുത്തിരിക്കുന്ന പട്ടികയിൽ വിട്ട ഭാഗം പൂരിപ്പിച്ച് സ്ട്രീം ഫങ്ഷനുകളെ താരതമ്യം ചെയ്യുക.

താരതമ്യ സൂചനകൾ	*കൺസോൾ ഫങ്ഷനുകൾ*	*സ്ട്രീം ഫങ്ഷനുകൾ*
ആവശ്യമായ ഹെഡർ ഫയലുകൾ	....	....
ഉപയോഗ രീതി	*ബ്രാക്കറ്റിനകത്ത് ആവശ്യമായ ഡാറ്റയോ, വേരിയബിളിന്റെ പേരോ ചേർത്ത് ഫങ്ഷന്റെ പേര് സൂചിപ്പിക്കുക*	*ഒബ്ജക്റ്റിന്റെ തുടർച്ചയായി ഡോട്ട് ഓപ്പറേറ്ററും, ആവശ്യമായ ഡാറ്റയോ വേരിയബിളിന്റെ പേരോ ചേർത്തിട്ടുള്ള ഫങ്ഷൻ ഉപയോഗിക്കുക.*
ഉപകരണങ്ങൾ	*കീബോർഡോ, മോണിറ്ററോ എന്ന് സൂചിപ്പിക്കപ്പെട്ടില്ലെങ്കിൽ*	
ഉദാഹരണങ്ങൾ	....	....

സ്വയം പരിശോധിക്കാം

1. കാരക്റ്റർ ഡാറ്റ ഇൻപുട്ട് ചെയ്യുന്നതിനുള്ള ഒരു സ്ട്രീം ഫങ്ഷന്റെ പേരെഴുതുക?
2. `write()` ഫങ്ഷൻ ഉപയോഗിച്ച് "Smoking is injurious to health"എന്ന സ്ട്രിങ്ങ് പ്രദർശിപ്പിക്കുവാനുള്ള C++ പ്രസ്താവന എഴുതുക?
3. സ്ട്രീം ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഫങ്ക്ഷനുകൾ ഉപയോഗിക്കുന്നതിനാവശ്യമായ ഹെഡർ ഫയലിന്റെ പേരെഴുതുക?
4. `getline()` ഫങ്ഷന്റെ വാക്യഘടന (syntax) എഴുതുക?

നമുക്ക് സംഗ്രഹിക്കാം

സ്ട്രിങ്ങുകളെ കൈകാര്യം ചെയ്യുന്നതിനാണ് C++ പ്രോഗ്രാമുകളിൽ കാരക്റ്റർ അറേ ഉപയോഗിക്കുന്നത്. ഒരു സ്ട്രിങ്ങിനു മെമ്മറി അനുവദിക്കപ്പെടുമ്പോൾ സ്ട്രിങ്ങിന്റെ അവസാന ഭാഗത്താണ് നൾ കാരക്റ്റർ ('\0') കൂട്ടിച്ചേർക്കപ്പെടുന്നത്. സ്ട്രിങ്ങുകൾ ഇൻപുട്ട്/ ഔട്ട്പുട്ട് ചെയ്യുന്നതിന് വിവിധ തരം കൺസോൾ ഫങ്ഷനുകൾ ലഭ്യമാണ്. ഇവ cstdio, എന്ന ഹെഡർ ഫയലിലാണ് ഉൾപ്പെട്ടിട്ടുള്ളത്. iostream എന്ന ഹെഡർ ഫയലും സ്ട്രിങ്ങുകളുടെ ഇൻപുട്ട്/ഔട്ട്പുട്ട് പ്രയോഗങ്ങൾക്കുള്ള ചില സ്ട്രീം ഫങ്ഷനുകൾ ലഭ്യമാക്കുന്നുണ്ട് .

പഠന നേട്ടങ്ങൾ

ഈ പാഠഭാഗത്തിന്റെ പൂർത്തീകരണത്തിനു ശേഷം പഠിതാവ്

- സ്ട്രിങ് കൈകാര്യം ചെയ്യുന്നതിന് കാരക്റ്റർ അറെ ഉപയോഗിക്കുന്നതിനുള്ള ശേഷി ആർജിക്കുന്നു.
- കാരക്റ്റർ, സ്ട്രിങ് എന്നിവ ഇൻപുട്ട്/ഔട്ട്പുട്ട് ചെയ്യുന്നതിനുള്ള വിവിധ തരം ബിൽറ്റ് ഇൻ ഫങ്ഷനുകൾ (Built in function) ഉപയോഗിക്കുന്നതിനുള്ള കഴിവ് നേടുന്നു.
- കൺസോൾ ഫങ്ഷനുകളും സ്ട്രീം ഫങ്ഷനുകളും താരതമ്യം ചെയ്യാനുള്ള പ്രാവീണ്യം നേടുന്നു.

ലാബ് പ്രവർത്തനങ്ങൾ

1 ഒരു സ്ട്രിങ് ഇൻപുട്ട് ചെയ്ത് അതിലെ വലിയ അക്ഷരങ്ങൾ, ചെറിയ അക്ഷരങ്ങൾ, സംഖ്യകൾ, പ്രത്യേക കാരക്റ്ററുകൾ, വൈറ്റ് സ്പേസുകൾ എന്നിവയുടെ എണ്ണം കണ്ടുപിടിക്കുന്നതിനുള്ള C++ പ്രോഗ്രാം എഴുതുക.

2. ഒരു വാചകത്തിലെ വാക്കുകളുടെ എണ്ണം കണ്ടെത്തുന്നതിനുള്ള C++ പ്രോഗ്രാം എഴുതുക.

3. ഒരു സ്ട്രിങ് ഇൻപുട്ട് ചെയ്ത് അതിലെ എല്ലാ ചെറിയ സ്വരാക്ഷരങ്ങളും (vowels) വലിയ സ്വരാക്ഷരങ്ങളാക്കി മാറ്റുന്നതിനുള്ള C++ പ്രോഗ്രാം എഴുതുക.

4. തന്നിരിക്കുന്ന സ്ട്രിങ്ങിനെ തിരിച്ചെഴുതുന്നതിനുള്ള പ്രോഗ്രാം എഴുതുക. ഉദാഹരണത്തിന് "AND" ആണ് ഇൻപുട്ട് ചെയ്തതെങ്കിൽ ഔട്ട്പുട്ടായി "DNA" എന്ന് ലഭിക്കണം.

5. ഇൻപുട്ട് ചെയ്ത വാക്ക് ഉപയോഗിച്ച് (ഉദാഹരണത്തിന് COMPUTER) താഴെ കാണുന്നവിധം ഒരു ത്രികോണം നിർമ്മിക്കുന്നതിനുള്ള പ്രോഗ്രാം എഴുതുക.

```
C
C    O
C    O    M
C    O    M    P
C    O    M    P    U
C    O    M    P    U    T
C    O    M    P    U    T    E
C    O    M    P    U    T    E    R
```

6 തന്നിരിക്കുന്ന വാചകത്തിലെ ഓരോ വാക്കിന്റെയും ആദ്യ കാരക്റ്റർ മാത്രം പ്രദർശിപ്പിക്കുവാനുള്ള പ്രോഗ്രാം എഴുതുക. ഇവിടെ കൺസോൾ ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഫങ്ഷനുകൾ മാത്രമേ ഉപയോഗിക്കാവൂ. ഉദാഹരണത്തിന് "Save Water Save Nature" എന്ന സ്ട്രിങ് ഇൻപുട്ട് ചെയ്താൽ ഔട്ട്പുട്ട് "SWSN" എന്നായിരിക്കണം.

7. ഒരു സ്ട്രിങ് പാലിൻഡ്രോം ആണോ അല്ലയോ എന്ന് പരിശോധിക്കുന്നതിനുള്ള പ്രോഗ്രാം എഴുതുക. (ഒരു സ്ട്രിങ്ങും അതിന്റെ തിരിച്ചെഴുതിയ രൂപവും ഒരേ പോലെയാണെങ്കിൽ ആ സ്ട്രിങ് പാലിൻഡ്രോം ആണ്. ഉദാഹരണം "MALAYALAM").

മാതൃക ചോദ്യങ്ങൾ

ഹ്രസ്വോത്തര ചോദ്യങ്ങൾ

1. putchar(97); എന്ന പ്രസ്താവനയുടെ ഔട്ട്പുട്ട് എന്തായിരിക്കും?
2. കൺസോൾ ഫങ്ഷന്റെയും സ്ട്രിങ് ഫങ്ഷന്റെയും വ്യത്യാസം എഴുതുക.
3. get() ഫങ്ഷനുപയോഗിച്ച് "computer" എന്ന സ്ട്രിങ് ഇൻപുട്ട് ചെയ്യുവാനുള്ള C++ പ്രസ്താവന എഴുതുക.
4. താഴെ കൊടുത്തിട്ടുള്ള കോഡിന്റെ ഔട്ട്പുട്ട് എഴുതുക.

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

ലഘു ഉപന്യാസം

1. താഴെ കൊടുത്തിരിക്കുന്ന C++ കോഡിന്റെ ഔട്ട്പുട്ട് എഴുതുക

```
char str[] = "Program";
for(int i=0; str[i]!='\0'; ++)
    {
    putchar(str[i]);
    putchar('_');
    }
```

2. താഴെ കൊടുത്തിരിക്കുന്ന C++ പ്രോഗ്രാമിലെ തെറ്റുകൾ കണ്ടെത്തി കാരണം വ്യക്തമാക്കുക .

```
#include<iostream.h>
using namespace std;
int main()
    {
    char ch, str[10];
    write("Enter a character ");
    ch=getchar();
    puts("Enter a string");
    cin.getline(str);
    cout<<"The data entered are " <<ch;
    putchar(str);
    }
```

3. താഴെ കൊടുത്തിട്ടുള്ള ഫങ്ഷനുകൾ നിരീക്ഷിക്കുക. അവ സാധുവാണെങ്കിൽ, പ്രവർത്തിക്കുമ്പോൾ എന്ത് സംഭവിക്കുന്നുവെന്ന് വിവരിക്കുക. അവ അസാധുവാണെങ്കിൽ കാരണം എഴുതുക (വേരിയബിൾ സാധുവാണെന്ന് സങ്കൽപ്പിക്കുക).

```
a) getchar(ch);            b)   gets(str[5]);
c) putchar("hello");       d)   cin.getline(name,20,'.');
e) cout.write("hello world",10);
```

4. താഴെ കൊടുത്തിരിക്കുന്ന പ്രസ്താവനകൾ വായിച്ച് ചോദ്യങ്ങൾക്ക് ഉത്തരമെഴുതുക.

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

- ഇൻപുട്ട് സ്ട്രിങ്ങ് "Sachin Tendulkar" എന്നാണെങ്കിൽ ഔട്ട്പുട്ട് എന്തായിരിക്കും? ന്യായീകരിക്കുക.
- തന്നിരിക്കുന്ന ഇൻപുട്ട് സ്ട്രിങ് തന്നെ ഔട്ട്പുട്ടായി ലഭിക്കുന്നതിന് പ്രസ്താവനയിൽ ആവശ്യമായ മാറ്റം വരുത്തുക .

ഉപന്യാസം

1. കൺസോൾ ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഫങ്ഷനുകൾ ഉദാഹരണസഹിതം വിവരിക്കുക.
2. സ്ട്രീം ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഫങ്ഷനുകൾ ഉദാഹരണസഹിതം വിവരിക്കുക.