C++ ծրագրավորման լեզու.

C++-ը բազմակողմանի և ամուր ծրագրավորման լեզու է, որը հայտնի է իր արդյունավետությամբ, կատարողականությամբ և օբյեկտի վրա հիմնված ծրագրավորման առանձնահատկություններով: Այն մշակվել է Bjarne Stroustrup-ի կողմից Bell Labs-ում 1980-ականների սկզբին՝ որպես C ծրագրավորման լեզվի ընդլայնում: C++-ը նախագծված էր C-ի ցածր մակարդակի հիշողության մանիպուլյացիայի հնարավորությունները համատեղելու համար բարձր մակարդակի աբստրակցիաների հետ՝ այն հարմարեցնելով համակարգային մակարդակի ծրագրավորման, ինչպես նաև հավելվածների մշակման համար:

C++-ը լայնորեն կիրառվում է տարբեր տիրույթներում, այդ թվում՝ համակարգային ծրագրային ապահովում, խաղերի մշակում, ներկառուցված համակարգեր, գիտական հաշվարկներ, ձեռնարկությունների ծրագրեր և այլն: Դրա բազմակողմանիությունը պայմանավորված է իր հարուստ գործառույթներով, ներառյալ օբյեկտի վրա հիմնված ծրագրավորումը, ձևանմուշները, ստանդարտ գրադարանները և հիշողության կառավարումը:

Հիմնական օպերատորները C++-ում:

1. **Arithmetic Operators**: Այս օպերատորները օգտագործվում են մաթեմատիկական գործողությունների համար, ներառյալ գումարումը (+), հանումը (-), բազմապատկումը (*), բաժանումը (/) և մոդուլը (%):
2. **Relational Operators**: Այս օպերատորները օգտագործվում են երկու արժեքներ համեմատելու համար: Դրանք ներառում են հավասար (==), ոչ հավասար (!=), փոքր (<), մեծ քան (>), փոքր կամ հավասար (<=) և մեծ կամ հավասար (>=):
3. **Logical Operators**: Տրամաբանական օպերատորները օգտագործվում են տրամաբանական գործողությունների համար: &&-ը տրամաբանական է AND, || տրամաբանական է ԿԱՄ, և ! տրամաբանական է ՈՉ.

4. **Assignment Operators**: Նշանակման օպերատորները օգտագործվում են արժեքներ նշանակելու համար: Հիմնական նշանակման օպերատորը = է, և կան բարդ նշանակման օպերատորներ, ինչպիսիք են +=, -=, *=, /= և %=:
5. **Bitwise Operators**: Bitwise օպերատորները կատարում են գործողություններ բիթ մակարդակով: Դրանք ներառում են bitwise AND (&), bitwise OR (|), bitwise XOR (^), bitwise NOT (~), ձախ հերթափոխ (<<) և աջ հերթափոխ (>>):
6. **Increment and Decrement Operators**: ++-ը ավելացման օպերատորն է, իսկ ---ը նվազեցման օպերատորն է:

Այս օպերատորները հիմնարար են C++ ծրագրի շրջանակներում տարբեր գործողություններ և հաշվարկներ կատարելու համար:

Կոդի բլոկի կառուցվածքը:

C++-ում կոդը կազմակերպվում է բլոկների կամ շրջանակների: C++ ծրագրի տիպիկ կառուցվածքը ներառում է ֆունկցիաների սահմանումներ և հիմնական ֆունկցիա, որը ծառայում է որպես ծրագրի մուտքի կետ:

Ահա մի օրինակ.

```
#include <iostream>
using namespace std;

int main() {
int x = 5;
cout<<x<<endl;
   return 0;
}
```

C++-ում ծրագիրը սովորաբար բաղկացած է մի քանի ֆունկցիաներից, իսկ main() ֆունկցիան այն է, որտեղ սկսվում է ծրագրի կատարումը:

Compiler աշխատանք:

C++ աղբյուրի կոդը գործարկվող ծրագրի վերածելու գործընթացը ներառում է մի քանի փուլ, որը կառավարվում է կոմպիլյատորի կողմից.

1. **Preprocessing**: Նախապրոցեսորը մշակում է հրահանգներ, ինչպիսիք են #include և #define: Այն մշակում է վերնագրի ֆայլերը և մակրոները՝ պատրաստելով կոդը կոմպիլյացիայի համար:
2. **Compilation**: Կազմողը թարգմանում է նախապես մշակված կոդը թիրախային ճարտարապետությանը հատուկ ցածր մակարդակի մեքենայի հրահանգների՝ ստեղծելով օբյեկտային ֆայլեր:
3. **Linking**: Կապակցիչը միավորում է բազմաթիվ օբյեկտային ֆայլեր և լուծում է ֆունկցիաների և փոփոխականների հղումները: Այն ստեղծում է վերջնական գործարկվող ֆայլը՝ կապելով անհրաժեշտ գրադարանները և այլ բաղադրիչները:
4. **Loading**: Օպերացիոն համակարգի բեռնիչը բեռնում է գործարկվող ֆայլը հիշողության մեջ՝ նախապատրաստելով այն կատարման:
5. **Execution**: CPU-ն մեկնաբանում և կատարում է հիշողության մեջ պահվող մեքենայի մակարդակի հրահանգները՝ գործարկելով ծրագիրը:

Օբյեկտ-կողմնորոշված ծրագրավորման (OOP) սկզբունքներ:

C++-ը հայտնի է օբյեկտի վրա հիմնված ծրագրավորման (OOP) սկզբունքներին իր ուժեղ աջակցությամբ, որոնք ներառում են.

1. **Classes and Objects**: C++-ը թույլ է տալիս սահմանել դասեր, որոնք օգտագործողի կողմից սահմանված տվյալների տեսակներն են, որոնք ամփոփում են տվյալներ և այդ տվյալների հետ կապված մեթոդներ (գործառույթներ): Օբյեկտները դասերի օրինակներ են:
2. **Inheritance**: Ժառանգությունը թույլ է տալիս դասին (ստացված դասին) ժառանգել հատկություններ և վարքագիծ մեկ այլ դասից (բազային դաս): Սա հեշտացնում է կոդի վերօգտագործումը և հիերարխիկ դասի կառուցվածքի ստեղծումը:

3. **Polymorphism**: Պոլիմորֆիզմը թույլ է տալիս ֆունկցիային կամ մեթոդին այլ կերպ վարվել՝ ելնելով այն օբյեկտի տեսակից, որով կոչվում է: Սա ձեռք է բերվում ֆունկցիաների գերբեռնվածության և վիրտուալ գործառույթների միջոցով:
4. **Encapsulation**: Էկապսուլյացիան տվյալների և մեթոդների միավորումն է, որոնք գործում են այդ տվյալների վրա մեկ միավորի (դասի) շրջանակներում: Այն ապահովում է տվյալների անվտանգություն՝ սահմանափակելով մուտքը դասի որոշակի անդամների:
5. **Abstraction**: Աբստրակցիան ներառում է իրականացման բարդ մանրամասները թաքցնելը և օբյեկտի միայն անհրաժեշտ հատկանիշների բացահայտումը: Այն օգնում է կենտրոնանալ այն բանի վրա, թե ինչ է անում օբյեկտը, այլ ոչ թե ինչպես է այն հասնում իր ֆունկցիոնալությանը:

Ստանդարտ ձևանմուշների գրադարան (STL):

Ստանդարտ ձևանմուշների գրադարանը (STL) ընդհանուր ալգորիթմների և տվյալների կառուցվածքների հզոր հավաքածու է, որը տրամադրվում է C++-ի կողմից: Այն ներառում է բեռնարկղեր (օրինակ՝ վեկտորներ, ցուցակներ, քարտեզներ), ալգորիթմներ (տեսակավորում, որոնում) և ֆունկցիայի օբյեկտներ: STL-ի ըմբռնումը և օգտագործումը կարևոր է արդյունավետ և պահպանվող C++ կոդի համար:

Հիշողության կառավարում:

C++-ն ապահովում է հիշողության ձեռքով կառավարում ցուցիչների միջոցով, որոնք թույլ են տալիս հիշողության հասցեների ուղղակի մանիպուլյացիա: Այնուամենայնիվ, դա կարող է հանգեցնել հիշողության արտահոսքի և այլ խնդիրների: Կարևոր է հասկանալ հիշողության բաշխումը (օգտագործելով նորը) և տեղաբաշխումը (օգտագործելով ջնջումը), ինչպես նաև լավագույն փորձը, ինչպիսիք են խելացի ցուցիչները և RAII (Resource Acquisition Is Initialization) օգտագործումը հիշողության ավելի կայուն կառավարման համար:

Բացառությունների բեռնաթափում:

Բացառությունների մշակումը C++-ում թույլ է տալիս նրբագեղ կառավարել սխալները կամ բացառիկ պայմանները, որոնք կարող են առաջանալ ծրագրի կատարման

ընթացքում: Այն ներառում է բացառություններ նետել՝ օգտագործելով նետելը և դրանք վարել՝ օգտագործելով փորձել, բռնել և նետել: Սխալների ճիշտ մշակումը չափազանց կարևոր է ամուր և հուսալի C++ ծրագրերի համար:

Ֆայլ I/O:

C++-ը տրամադրում է տարբեր մեխանիզմներ ֆայլերից կարդալու և դրանց գրելու համար: Ֆայլերի մուտքագրման/ելքի գործառնությունների (I/O) և ֆայլերի հոսքերի (ifstream, ofstream և fstream) հասկանալը կարևոր է ֆայլերում պահվող տվյալների մշակման համար:

Multi-Threading և Concurrency:

C++-ն աջակցում է բազմաշերտ և միաժամանակյա ծրագրավորում, ինչը հնարավորություն է տալիս միաժամանակ մի քանի թելեր կատարել: Թեմաների ստեղծման, համաժամացման և համակարգման հասկանալը կարևոր է արդյունավետ և միաժամանակյա հավելվածներ ստեղծելու համար:

Կատարման օպտիմիզացում:

C++ կոդի օպտիմալացման մեթոդների ըմբռնումը, ինչպիսիք են հիշողության բաշխումը նվազագույնի հասցնելը, արդյունավետ ալգորիթմների և տվյալների կառուցվածքների օգտագործումը և կատարողականի բարելավման համար պրոֆիլավորումը, կենսական նշանակություն ունեն բարձր արդյունավետությամբ հավելվածներ մշակելու համար:

Այս թեմաները ձեր փաստաթղթում ներառելը կապահովի C++ ծրագրավորման լեզվի և դրա էական հատկանիշների և լավագույն փորձի համապարփակ ակնարկ: