Typedef, Struct e Ponteiros em C

Operadores de ponteiro

- Operador *: faz acesso ao conteúdo de uma área de memória indicada por um ponteiro.
- Operador &: retorna o ponteiro (endereço) de uma variável.

Declaração de uma variável ponteiro

tipo **nome_da_variável*;

int *px;

Exemplo

```
#include <iostream>

using namespace std;

int main()
{
   int x, *px, y, *py;
   px = &x;
   py = &y;
   cout << "Digite X: ";
   cin >> x;
   cout << "Digite Y: ";
   cin >> y;
   cout << "X e PX = " << x << " " << *px << endl;
   cout << "Y e PY = " << y << " " << *py << endl;
   return 0;
}
```

Funções

```
#include <iostream>
using namespace std;
int soma(int a, int b) {
   return a+b;
}
int main()
{
   int x, y, z;
   cout << "Digite X: ";
   cin >> x;
   cout << "Digite Y: ";
   cin >> y;
   z = soma(x, y);
   cout << "Z = " << z << endl;
   return 0;
}
```

```
#include <iostream>
using namespace std;
void troca(int *a, int *b) {
   int aux;
   aux = *a;
   *a = *b;
   *b = aux;
}
int main()
{
   int x, y;
   cout << "Digite X: ";
   cin >> x;
   cout << "Digite Y: ";
   cin >> y;
   troca(&x, &y);
   cout << "X = " << x << endl;
   cout << "Y = " << y << endl;
   return 0;
}
```

Estruturas

Uma estrutura (struct) é um agrupamento de variáveis em um registro único. Registro é um tipo de variável que pode suportar um número fixo de elementos de tipos diferentes. Cada elemento é chamado de campo e deve ter um nome associado a ele.

```
#include <iostream>

using namespace std;
```

```
int main()
{
   struct data {
      int dia, mes, ano;
   };
   struct pessoa {
      char nome[50];
      double salario;
      data nasc;
   };
   pessoa p;
   cout << "Nome: ";
   gets(p.nome);
   cout << "Salario: ";
   cin >> p.salario;
   cout << "Dia: ";
   cin >> p.nasc.dia;
   cout << "Mes: ";
   cin >> p.nasc.mes;
   cout << "Ano: ";
   cin >> p.nasc.ano;
   cout << "Nome = " << p.nome << endl;
   cout << "Salario = " << p.salario << endl;
   cout << "Nasc = " << p.nasc.dia << "/" << p.nasc.mes << "/" << p.nasc.ano << endl;
   return 0;
}
```

Ponteiro para Estrutura

```
#include <iostream>

using namespace std;
```

```
int main()
{
   struct data {
      int dia, mes, ano;
   };
   struct pessoa {
      char nome[50];
      double salario;
      data nasc;
   };
   pessoa p, *pp;
   cout << "Nome: ";
   gets(p.nome);
   cout << "Salario: ";
   cin >> p.salario;
   cout << "Dia: ";
   cin >> p.nasc.dia;
   cout << "Mes: ";
   cin >> p.nasc.mes;
   cout << "Ano: ";
   cin >> p.nasc.ano;
   pp = &p;
   cout << "Nome = " << (*pp).nome << endl;
   cout << "Salario = " << pp->salario << endl;
   cout << "Nascimento = "
        << pp->nasc.dia << "/"
        << pp->nasc.mes << "/"
        << pp->nasc.ano
        << endl;
   return 0;
}
```

Definição de Tipo

```
#include <iostream>
using namespace std;
typedef int inteiro;
typedef char cadeia[50];
int main()
{
   inteiro n;
   cadeia nome;
   cout << "Numero: ";
   cin >> n;
   cout << "Nome: ";
   cin >> nome;
   cout << "Numero = " << n << endl;
   cout << "Nome = " << nome << endl;
   return 0;
}
```

```
#include <iostream>
using namespace std;
typedef struct {
   int dia, mes, ano;
} data;

int main()
{
   data nasc, casam;
   cout << "Nasc Dia: ";
   cin >> nasc.dia;
   cout << "Nasc Mes: ";
   cin >> nasc.mes;
   cout << "Nasc Ano: ";
   cin >> nasc.ano;
```

```
   cout << "Casam Dia: ";
   cin >> casam.dia;
   cout << "Casam Mes: ";
   cin >> casam.mes;
   cout << "Casam Ano: ";
   cin >> casam.ano;
   cout << "Nasc = " << nasc.dia << "/" << nasc.mes << "/" << nasc.ano << endl;
   cout << "Casam = " << casam.dia << "/" << casam.mes << "/" << casam.ano << endl;
   return 0;
}
```

Exercício

Observe o programa a seguir mostrando o tratamento de números complexos.

```
#include <iostream>

using namespace std;

#define QTDNC 2

typedef struct {
   float real, imaginaria;
} numcomplex;

void leNumComplex(numcomplex *nc) {
   cout << endl << "Parte Real: ";
   cin >> nc->real;
   cout << endl << "Parte Imaginaria: ";
   cin >> nc->imaginaria;
}
```

```
void escreveNumComplex(numcomplex nc) {
   cout << endl << "Numero Complexo: ";
   if ((nc.real != 0) || (nc.imaginaria == 0)) {
      cout << nc.real;
   }
   if (nc.imaginaria != 0) {
      if ((nc.imaginaria > 0) && (nc.real != 0)) {
         cout << "+" << nc.imaginaria;
      } else {
         cout << nc.imaginaria;
      }
      cout << "i";
   }
   cout << endl;
}

numcomplex somaNumComplex(numcomplex a, numcomplex b) {
}
numcomplex produtoNumComplex(numcomplex a, numcomplex b) {
}

int main()
{
   int i;
   numcomplex vetorNC[QTDNC];
   numcomplex sNC, pNC;
   for (i=0; i<QTDNC; i++) {
      cout << endl << "====================";
      cout << endl << i+1 << "o Numero Complexo";
      cout << endl << "====================" << endl;
      leNumComplex(&vetorNC[i]);
      escreveNumComplex(vetorNC[i]);
   }
   cout << endl << "====================" << endl;
```

```
    cout << endl << "====================";
    cout << endl << "Soma";
    cout << endl << "====================" << endl;
    sNC = somaNumComplex(vetorNC[0], vetorNC[1]);
    escreveNumComplex(sNC);
    cout << endl << "====================" << endl;
    cout << endl << "====================";
    cout << endl << "Produto";
    cout << endl << "====================" << endl;
    pNC = produtoNumComplex(vetorNC[0], vetorNC[1]);
    escreveNumComplex(pNC);
    cout << endl << "====================" << endl;
    return 0;
}
```

a) Faça as rotinas somaNumComplex e produtoNumComplex para o cálculo da soma de dois números complexos e para o produto de dois números complexos respectivamente.

b) Faça novas rotinas para:

- Igualdade de dois números complexos.
- Oposto de um número complexo.
- Conjugado de um número complexo.