C++プログラミング講習

変数

1. 変数

変数は、データを保存しておくメモリ領域のことです。

データというのは、ゲームで例えるならば、

整数のHP,攻撃力,スコア

文字列のプレイヤー名

とかですかね…

簡単に言うと、変数はメモです。

変数には名前をつけることができ、変数名といいます。

ここでは、HPが変数名になっています。

変数のデータを値と呼び、ここでは、100がHPという変数の値です。

ではintは何でしょうか?

intは変数の型の一種です。

型は保存できるデータの種類です。さっきの例だと整数や文字列といったものが型です。

C++の様々な型

|  |  |
| --- | --- |
| C++での型 | 日本語 |
| int | 整数型(正確には32bit符号なし整数型) |
| float | 実数型(正確には単精度浮動小数点数型) |
| string | 文字列型 |
| char | 文字型 |
| bool | 真理値型 |

など…

ここで文字列と文字の違いを説明すると、

文字はaとかbとかの記号としての文字のことで、

文字列は、文字を複数集めたものです。

なぜ分ける必要があるのかはおまけに書くかも(書きました)

C++などの静的型付け言語では変数を宣言するときに型を指定する必要があります。

では、C++で変数を使ってみましょう

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #include <bits/stdc++.h>  using namespace std;  int main() {  int A;  A = 10;  cout << A + 10 << endl;  int B = 100;  B = B + 3;  cout << B << endl;  } |

上から見ていきます。

変数を使うとき、はじめに宣言を行う必要があります。

宣言は、型を書いた後に変数名を書くことでできます。

例:

|  |  |
| --- | --- |
| 1 | int name; |

6行目では、int型で変数名がAの変数を宣言しています。

変数に値を上書きするとき、代入を使います。つまり、前の値を消してから、新しい値を書き込みます。

例:

|  |  |
| --- | --- |
| 1 | name = 100; |

注意⚠この代入の=は数学の等号(例:3+4=7)とは全く別物なので気をつけましょう

7行目では、変数Aに10を代入しています。

変数の値を見たいときは、参照をします。変数名を書くだけです。

例:

|  |  |
| --- | --- |
| 1 | cout << name << endl; |

10行目 変数を宣言と同時に代入を行うこともできます。初期化とも言います。

ここでクイズ

12行目の出力はどうなるでしょうか。正解は[103]

なぜそうなるかを見てみましょう。

|  |  |
| --- | --- |
| 1  2 | int B = 100;  B = B + 3; |

実はこれはさっき１章ででてきた演算子の優先順位が関わっています。

プログラムは基本上から下、左から右に実行されますが、演算子には、優先順位があることを、１章でやりました。そして代入=の優先順位は今まで見てきた演算子の中で、最も優先順位が低い演算子です。なので、

|  |  |
| --- | --- |
| 1  2 | int B = 100;  B = (B + 3);//←この順番 |

B + 3が実行されてから、B + 3の値がBに代入されます。

[参考] [CとC++の演算子-演算子の優先順位](https://ja.wikipedia.org/w/index.php?title=C%E3%81%A8C%2B%2B%E3%81%AE%E6%BC%94%E7%AE%97%E5%AD%90#%E6%BC%94%E7%AE%97%E5%AD%90%E3%81%AE%E5%84%AA%E5%85%88%E9%A0%86%E4%BD%8D)

このコードはこのように書くこともできます

|  |  |
| --- | --- |
| 1  2 | int B = 100;  B += 3; |

優先順位は代入=と同じで、他の四則演算もできます。 += -= \*= /= %=

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | #include <bits/stdc++.h>  using namespace std;  int main() {  int A = 10, B;  B = A;  A = 100;  cout << A << endl;  cout << B << endl;  } |

5行目 カンマ , で区切ることで、一度に同じ型の変数を複数宣言できます。

6行目では、BにAの値をコピーしています。

コピーした後、Aの値が変わってもBは影響を受けません。

変数名のルール

・数字で始まる名前

・半角のアンダーバー \_ 以外の記号が入っている名前

・キーワード(すでにC++で使われている単語 例:型名)[参考]:[キーワード (C++)](https://ja.wikipedia.org/wiki/%E3%82%AD%E3%83%BC%E3%83%AF%E3%83%BC%E3%83%89_(C%2B%2B))

・同じ変数名(スコープが違う場合を除く)

型変換

異なる型同士の計算では、型変換が行われます。

できない方同士は、コンパイルエラーになります。(例: 文字列と数字)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | #include <bits/stdc++.h>  using namespace std;  int main() {  int A = 10;  float pie = 3.14;  string S = "FFT";  cout << (int)pie << endl;  cout << pie \* A \* A << endl;  cout << 10 + pie << endl;  cout << A / pie << endl;  pie = A;  cout << pie << endl;  /\*  ↓コンパイルエラーになる処理  cout << S + A << endl;  cout << pie \* A << endl;  S = A;  \*/  } |

浮動小数点数の小数部の桁数を指定したいときは出力する前にこれをしてください。

|  |  |
| --- | --- |
| 1 | cout << fixed << setprecision(15);　//15桁に設定 |

8行目では、計算せずに、ただ型の変換だけを行っていて、そのような型変換を

キャストと呼びます。浮動小数点数型の数を整数型の数にキャストすると、小数点以下の値が切り捨てられます。

入力を受け取る

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #include <bits/stdc++.h>  using namespace std;  int main() {  int a; string s;  cin >> a >> s;  cout << a << endl << s << endl;  } |

cinと>>を使います。変数に入力の値をいれることができます。

変数のスコープ

main関数は、{ }で囲われています。この{ }で囲われている部分をブロックと言います。

ブロック内で定義された変数は、それより内側のブロックでしか使えないというルールがあります。また、その変数が使える範囲をスコープといいます

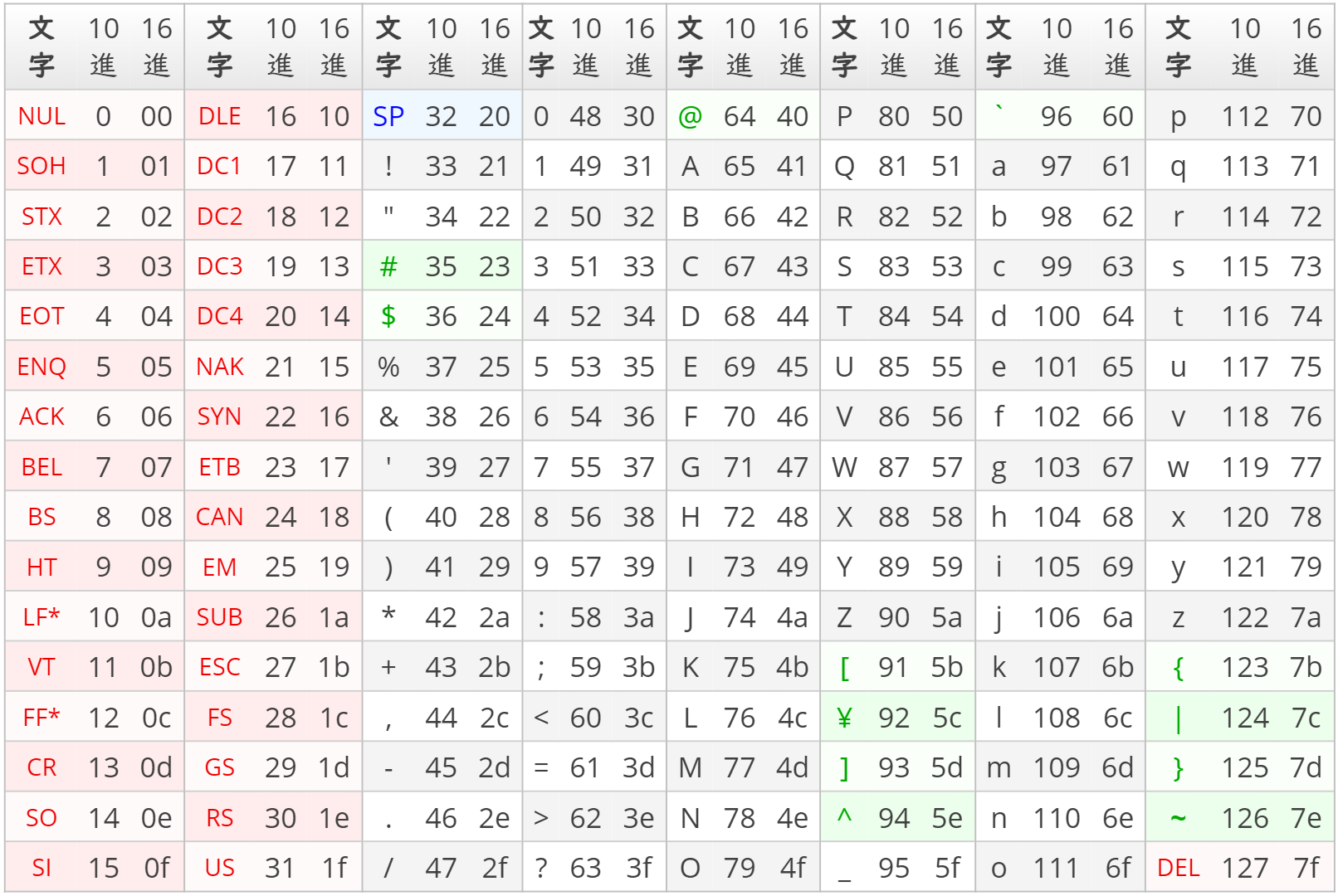
同じ名前の変数が複数ある場合、最も内側で定義された変数を使います。(同じブロック内で同じ変数を定義することはできない)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #include <bits/stdc++.h>  using namespace std;  int main() {  int A = 10, B = 3;  {  string A = "HELLO", C = "CPP";  cout << A << endl; //より内側で定義された方を使う  cout << B << endl;  }  //cout << C << endl; ←Cをスコープ外で参照しているのでコンパイルエラー  cout << A << endl;  cout << B << endl;  } |

また、関数の外部で変数を定義することもでき、そのような変数をグローバル変数と呼びます。

おまけ

コンピューターで半角文字を扱うとき、ASCIIコードがよく使われています。

C++もそうで、文字には、ASCIIコードに対応した数字が振り分けられています。

([e-words.jp/w/ASCII.html](file:///C:\Users\USER\OneDrive\ドキュメント\e-words.jp\w\ASCII.html)より)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #include <bits/stdc++.h>  using namespace std;  int main() {  char C = 'a';  cout << (int)C << endl;  } |

このようにchar型の変数を整数型に型変換させると、その文字に対応する10進数のASCIIコードの値が見られます。

またアルファベットは、連番になっているため、このような足し算引き算ができます。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #include <bits/stdc++.h>  using namespace std;  int main() {  char C = 'A';  cout << 'D' - C << endl;  C += 1;  cout << C << endl;  cout << C + 1 << endl;  cout << char(C + 1) << endl;  cout << char(65) << endl;  } |

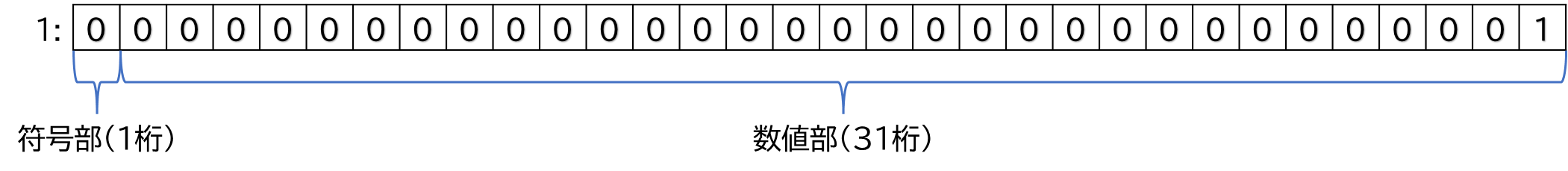
初期化する前の変数

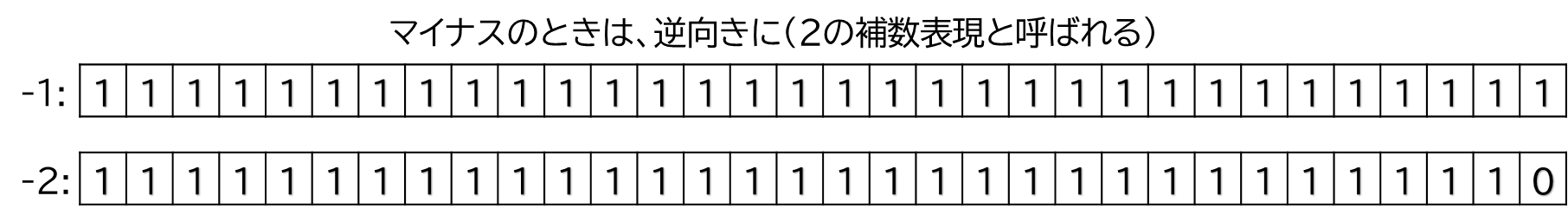
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #include <bits/stdc++.h>  using namespace std;  int main() {  int a;  cout << a << endl;  } |

実行してみると、環境によっては、変な値が出てきたり、0になったり、コンパイルエラーになったりします。初期化していないローカル変数を読み取ると、未定義動作を引き起こします。未定義動作については[未定義動作-wikipedia](https://ja.wikipedia.org/wiki/%E6%9C%AA%E5%AE%9A%E7%BE%A9%E5%8B%95%E4%BD%9C)参考

未定義動作は、なにが起こるのかがわからないのでなるべくやらないように気をつけましょう。

オーバーフロー 参考:[【2038年問題】世界中のシステムがぶっ壊れる？](https://www.youtube.com/watch?v=WLyIPslm_rw)

３２bit符号付き整数型であるint型は名前の通り、メモリ内部では３２桁の２進数で表されています

正の整数のときは、符号部は０で、負の整数のときは、符号部を１にしています。

つまり、int型は-(2を32回かけた数)～(2を32回かけた数)－１までの範囲の数

(-2147483648～2147483647)しか表現できません。この範囲を超えてしまうことをオーバーフローと言い、オーバーフローすると正しい計算結果が求められません。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #include <bits/stdc++.h>  using namespace std;  int main() {  int i = 2000000000;  cout << i + 2000000000 << endl;  cout << (i + 2000000000) / 100 << endl;//計算途中に超えるのもだめ  } |

これを回避するためには、int64\_t型を使います。

int64\_t型は、64bit符号付き整数型で

-9223372036854775808～9223372036854775807を表せます。

intをint64\_tに変えて実行すると、正しい結果になりました。

ちなみにC++では符号つき整数型のオーバーフローは未定義動作です。